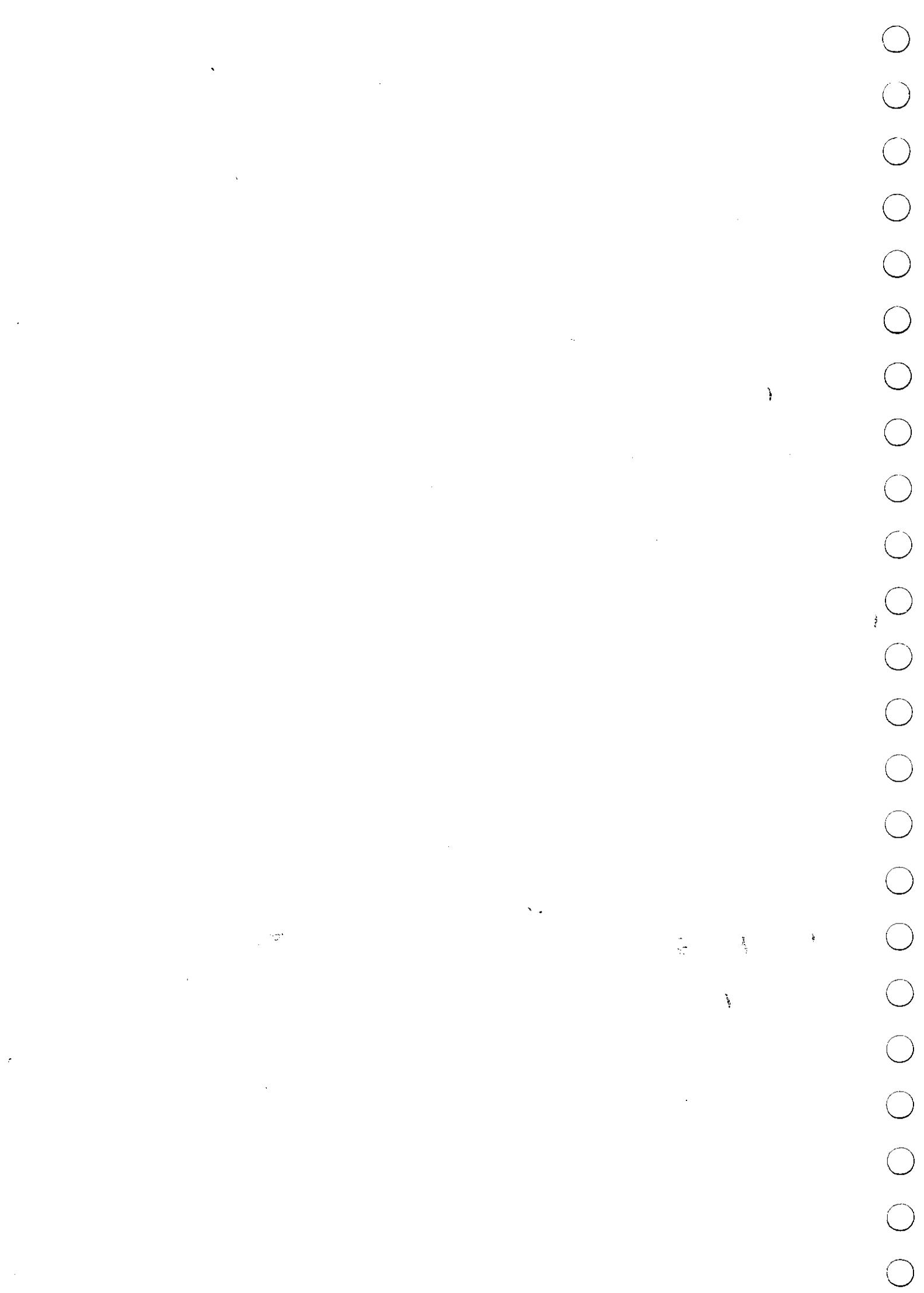




**RH134**



**Red Hat System Administration II  
Student Workbook  
Red Hat Enterprise Linux 6  
Release en-2-20110131**

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

# **RED HAT SYSTEM ADMINISTRATION II**

**Red Hat Enterprise Linux 6 RH135  
Red Hat System Administration II  
Edition 2**

Author	Joshua Hoffman
Author	Bowe Strickland
Author	Brad Smith
Author	Robert Locke
Author	George Hacker
Editor	Steven Bonneville
Editor	Mark Howson

Copyright © 2011 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are Copyright © 2011 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed please e-mail [training@redhat.com](mailto:training@redhat.com) or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, Hibernate, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a registered trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

All other trademarks are the property of their respective owners.

---

Contributors: Brian Butler, Victor Costea, Andrew Dingman

---

<b>Document Conventions</b>	vii
Notes and Warnings .....	vii
<b>Introduction</b>	ix
Welcome to class! .....	ix
About Red Hat Enterprise Linux .....	ix
Additional Red Hat Enterprise Linux Software .....	x
Contacting Red Hat Technical Support .....	xii
<b>About This Course</b>	xv
Red Hat System Administration II .....	xv
Structure of the Course .....	xv
Orientation to the Classroom Network .....	xvi
<b>Internationalization</b>	xvii
Language Support .....	xvii
System-wide Default Language .....	xvii
Per-user Language Selection .....	xvii
Input Methods .....	xviii
Language Codes Reference .....	xviii
<b>1. Automated Installation of Red Hat Enterprise Linux</b>	1
Introductory Overview .....	2
Create a Kickstart File with system-config-kickstart .....	3
Make the Kickstart File Available to Installers .....	5
Create Boot Media .....	7
Point the Installer to a Kickstart File .....	8
Modify a Kickstart File .....	11
Criterion Test .....	14
<b>2. Accessing the Command-line</b>	17
Accessing the Command-line Locally .....	18
Accessing the Command-line Using ssh .....	21
Criterion Test .....	24
<b>3. Intermediate Command-line Tools</b>	27
Using Hard Links .....	28
Archives and Compression .....	33
Introduction to vim .....	36
Basic vim Workflow .....	38
Criterion Test .....	40
<b>4. Regular Expressions, Pipelines and I/O Redirection</b>	43
Basic Regular Expressions .....	44
Using grep .....	49
Pipelines and Redirection .....	51
Criterion Test .....	55
<b>5. Network Configuration and Troubleshooting</b>	59
Understanding Network Configuration Files .....	60
Basic Troubleshooting Process .....	66
Network Troubleshooting Toolkit .....	68
Criterion Test .....	71
<b>6. Managing Simple Partitions and File Systems</b>	75

---

Simple Partitions and File Systems .....	76
Enabling Data Privacy with Partition Encryption .....	80
Managing Swap Space .....	84
Criterion Test .....	87
<b>7. Managing Flexible Storage with Logical Volume Manager</b>	<b>91</b>
Recognize the Components of LVM .....	92
Implement LVM Storage with Command-line Tools .....	96
Extend a Logical Volume and Ext4 File System .....	99
Extending and Reducing a Volume Group .....	103
Create a Snapshot to Facilitate Data Backup .....	105
Criterion Test .....	108
<b>8. Accessing Network File Sharing Services</b>	<b>111</b>
Mount Network File Systems .....	112
Automatically Mount Network Storage .....	117
Criterion Test .....	121
<b>9. Managing User Accounts</b>	<b>125</b>
What Is a User? .....	126
Managing Local Users .....	128
Managing Passwords .....	131
Criterion Test .....	134
<b>10. Network User Accounts with LDAP</b>	<b>137</b>
Network Authentication Using an LDAP Server .....	138
Network Mounting Home Directories .....	142
Criterion Test .....	145
<b>11. Controlling Access to Files</b>	<b>149</b>
Managing Groups .....	150
Managing File System Access Control Lists .....	153
Criterion Test .....	157
<b>12. Managing SELinux</b>	<b>161</b>
Basic SELinux Security Concepts .....	162
SELinux Modes .....	165
Display and Modify SELinux Modes .....	168
Display and Modify SELinux File Contexts .....	170
Managing SELinux Booleans .....	173
Monitoring SELinux Violations .....	174
Criterion Test .....	177
<b>13. Installing and Managing Software</b>	<b>181</b>
Using yum .....	182
Managing YUM Component Groups .....	185
Handling Third-Party Software .....	187
Using Third-Party yum Repositories .....	191
Criterion Test .....	195
<b>14. Managing Installed Services</b>	<b>199</b>
Manage Services .....	200
Confirm Service Availability .....	202
Criterion Test .....	204

---

<b>15. Analyzing and Storing Logs</b>	<b>207</b>
Determine Log Destinations .....	208
Locate and Analyze a Log Summary Report .....	212
Change the Log Summary e-mail Address .....	214
Criterion Test .....	217
<b>16. Managing Processes</b>	<b>221</b>
Monitoring Processes .....	222
Terminating and Governing Processes .....	223
Managing Periodic Tasks .....	227
Scheduling Deferred Tasks .....	230
Criterion Test .....	233
<b>17. Tuning and Maintaining the Kernel</b>	<b>237</b>
Supported Architectures and Kernel Identification .....	238
Managing Kernel Modules .....	240
Specifying Kernel Boot Parameters .....	242
Upgrading Your Kernel .....	244
<b>18. System Recovery Techniques</b>	<b>249</b>
The Boot Process .....	250
Repairing Boot Issues .....	258
Criterion Test .....	263
<b>A. Solutions</b>	<b>267</b>
Automated Installation of Red Hat Enterprise Linux .....	267
Accessing the Command-line .....	274
Intermediate Command-line Tools .....	278
Regular Expressions, Pipelines and I/O Redirection .....	283
Network Configuration and Troubleshooting .....	288
Managing Simple Partitions and File Systems .....	292
Managing Flexible Storage with Logical Volume Manager .....	301
Accessing Network File Sharing Services .....	310
Managing User Accounts .....	313
Network User Accounts with LDAP .....	316
Controlling Access to Files .....	320
Managing SELinux .....	324
Installing and Managing Software .....	329
Managing Installed Services .....	334
Analyzing and Storing Logs .....	338
Managing Processes .....	341
Tuning and Maintaining the Kernel .....	347
System Recovery Techniques .....	349



---

# Document Conventions

## Notes and Warnings



### Note

"Notes" are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.



### Comparison

"Comparisons" look at similarities and differences between the technology or topic being discussed and similar technologies or topics in other operating systems or environments.



### References

"References" describe where to find external documentation relevant to a subject.



### Important

"Important" boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled "Important" will not cause data loss, but may cause irritation and frustration.



### Warning

"Warnings" should not be ignored. Ignoring warnings will most likely cause data loss.



---

# Introduction

## Welcome to class!

Thank you for attending this Red Hat training class. Please let us know if you have any special needs while at our training facility.

Please ask the instructor if you have any questions about the facility, such as operating hours of the facility and when you will have access to the classroom, locations of restrooms and break rooms, availability of telephones and network connectivity, and information about the local area.

As a courtesy to other students, please place your pager or cell phone's ringer on vibrate or mute, or turn off your devices during class. We ask that you only make calls during break periods.

If you have a personal emergency and are unable to attend or complete the class, please let us know. Thank you!

## About Red Hat Enterprise Linux

This course is taught using Red Hat Enterprise Linux, an enterprise-targeted Linux distribution focused on mature open source software designed specifically for organizations using Linux in production settings.

Red Hat Enterprise Linux is sold on a subscription basis, where the subscription gives you continues access to all supported versions of the operating system in binary and source form, not just the latest one, including all updates and bug fixes. Extensive support services are included: a support contract and Update Module entitlement to Red Hat Network are included for the subscription period. Various Service Level Agreements are available that may provide up to 24x7 coverage with a guaranteed one hour response time for Severity 1 issues. Support will be available for up to seven years after a particular major release (ten years with the optional "Extended Update Support" Add-On).

Red Hat Enterprise Linux is released on a multi-year cycle between major releases. Minor updates to major releases are released roughly every six months during the lifecycle of the product. Systems certified on one minor update of a major release continue to be certified for future minor updates of the major release. A core set of shared libraries have APIs and ABIs which will be preserved between major releases. Many other shared libraries are provided, which have APIs and ABIs which are guaranteed within a major release (for all minor updates) but which are not guaranteed to be stable across major releases.

Red Hat Enterprise Linux is based on code developed by the open source community, which is often first packaged through the Red Hat sponsored, freely-available Fedora distribution (<http://fedoraproject.org/>). Red Hat then adds performance enhancements, intensive testing, and certification on products produced by top independent software and hardware vendors. Red Hat Enterprise Linux provides a high degree of standardization through its support for four processor architectures (32-bit Intel x86-compatible, AMD64/Intel 64 (x86-64), IBM POWER, and IBM mainframe on System z). Furthermore, we support the 4000+ ISV certifications on Red Hat Enterprise Linux whether the RHEL operating system those applications are using

is running on "bare metal", in a virtual machine, as a software appliance, or in the cloud using technologies such as Amazon EC2.

Currently, the Red Hat Enterprise Linux product family includes:

- *Red Hat Enterprise Linux for Servers*: the datacenter platform for mission-critical servers running Red Hat Enterprise Linux. This product includes support for the largest x86-64 and x86-compatible servers and the highest levels of technical support, deployable on bare metal, as a guest on the major hypervisors, or in the cloud. Subscriptions are available with flexible guest entitlements of one, four, or unlimited guests per physical host. Pricing is based on the basis of the number of socket-pairs populated on the system motherboard, the number of guests supported, the level of support desired, and the length of subscription desired.

*Red Hat Enterprise Linux for IBM POWER* and *Red Hat Enterprise Linux for IBM System z* are similar variants intended for those system architectures.

- *Red Hat Enterprise Linux Desktop*: built for the administrator and end-user, Red Hat Enterprise Linux Desktop provides an attractive and highly productive environment for knowledge workers on desktops and laptops. Client installations can be finely tailored and locked down for simplicity and security for any workstation task.

The basic *Desktop* variant is designed for task workers who have a limited amount of administrative control over the system, who primarily use productivity applications like Firefox Evolution/Thunderbird, OpenOffice.org, and Planner/TaskJuggler. The more sophisticated *Workstation* variant is designed for advanced Linux users who need a stand-alone development environment, and who are expected to have local super-user privileges or selected super-user privileges.

In addition, other variants exist such as *Red Hat Enterprise Linux for HPC Head Node* and *Red Hat Enterprise Linux for HPC Compute Node* (targeted at high-performance computing clusters), and *Red Hat Enterprise Linux for SAP Business Applications*. For more information please visit <http://www.redhat.com/>.

## Additional Red Hat Enterprise Linux Software

Two additional software update channels are provided with Red Hat Enterprise Linux beyond the core software packages shipped:

- *Supplementary*: the "Supplementary" channel provides selected closed source packages, built for Red Hat Enterprise Linux as a convenience to the customer. These include things like Adobe Flash or proprietary Java JVMs.
- *Optional*: the "Optional" channel provides selected open source packages, as a convenience only. They are generally included in another Red Hat Enterprise Linux variant as a fully-supported package, or are a build requirement for the distribution. These packages are only available through a Red Hat Network child channel.



## Important

*Supplementary and Optional* packages are provided with limited support, as a customer convenience only.

Red Hat also offers a portfolio of fully-supported *Add-Ons for Red Hat Enterprise Linux* which extend the features of your Red Hat Enterprise Linux subscription. These add-ons allow you to add capabilities and tailor your computing environment to your particular needs. These Add-Ons include support for high availability application clustering, cluster file systems and very large file systems, enhanced system management with Red Hat Network, extended update support, and more.



## Note

Please visit <http://www.redhat.com/rhel/add-ons/> for more information about available *Add-Ons for Red Hat Enterprise Linux*.

For information about other products which are provided by Red Hat, such as Red Hat Enterprise Virtualization, JBoss Enterprise Middleware, Red Hat Enterprise MRG, and various custom consulting and engineering services, <http://www.redhat.com/products/> also has useful information.

The Fedora Project also provides additional packages for Red Hat Enterprise Linux through *EPEL* (*Extra Packages for Enterprise Linux*). EPEL is a volunteer-based community effort to create a repository of high-quality add-on packages which can be used with Red Hat Enterprise Linux and compatible derivatives. It accepts legally-unencumbered free and open source software which does not conflict with packages in Red Hat Enterprise Linux or Red Hat add-on products. EPEL packages are built for a particular major release of Red Hat Enterprise Linux and will be updated by EPEL for the standard support lifetime of that major release.

Red Hat does not provide commercial support or service level agreements for EPEL packages. While not supported officially by Red Hat, EPEL provides a useful way to reduce support costs for unsupported packages which your enterprise wishes to use with Red Hat Enterprise Linux. EPEL allows you to distribute support work you would need to do by yourself across other organizations which share your desire to use this open source software in RHEL. The software packages themselves go through the same review process as Fedora packages, meaning that experienced Linux developers have examined the packages for issues. As EPEL does not replace or conflict with software packages shipped in RHEL, you can use EPEL with confidence that it will not cause problems with your normal software packages.

For developers who wish to see their open source software become part of Red Hat Enterprise Linux, often a first stage is to sponsor it in EPEL so that RHEL users have the opportunity to use it, and so experience is gained with managing the package for a Red Hat distribution.

Visit <http://fedoraproject.org/wiki/EPEL/> for more information about EPEL.



## Important

EPEL is supported by the community-managed Fedora Project and not by Red Hat Support.

# Contacting Red Hat Technical Support

One of the benefits of your subscription to Red Hat Enterprise Linux is access to technical support through Red Hat's customer portal at <http://access.redhat.com/>. If you do not have a Red Hat account on the customer portal or are not able to log in, you can go to <https://access.redhat.com/support/faq/LoginAssistance.html> or contact Customer Service for assistance.

You may be able to resolve your problem without formal technical support by searching Knowledgebase (<https://access.redhat.com/kb/knowledgebase/>). Otherwise, Red Hat Support may be contacted through a web form or by phone depending on your support level. Phone numbers and business hours for different regions vary; see <https://access.redhat.com/support/contact/technicalSupport.html> for current information. Information about the support process is available at [https://access.redhat.com/support/policy/support\\_process.html](https://access.redhat.com/support/policy/support_process.html).

Some tips on preparing your bug report to most effectively engage Red Hat Support:

- *Define the problem.* Make certain that you can articulate the problem and its symptoms before you contact Red Hat. Be as specific as possible, and detail the steps you can use (if any) to reproduce the problem.
- *Gather background information.* What version of our software are you running? Are you using the latest update? What steps led to the failure? Can the problem be recreated and what steps are required? Have any recent changes been made that could have triggered the issue? Were messages or other diagnostic messages issued? What exactly were they (exact wording may be critical)?
- *Gather relevant diagnostic information.* Be ready to provide as much relevant information as possible; logs, core dumps, traces, the output of **sosreport**, etc. Technical Support can assist you in determining what is relevant.
- *Determine the Severity Level of your issue.* Red Hat uses a four-level scale to indicate the criticality of issues; criteria may be found at [https://access.redhat.com/support/policy/GSS\\_severity.html](https://access.redhat.com/support/policy/GSS_severity.html).



## Warning

*Bugzilla is not a support tool!* For support issues affecting Red Hat Enterprise Linux, customers should file their bugs through the support channels discussed above in order to ensure that Red Hat is fully aware of your issue and can respond under the terms of your Service Level Agreement. Customers should *not* file bugs directly in the <http://bugzilla.redhat.com/> web interface.

For Red Hat Enterprise Linux, Bugzilla is used by engineering to track issues and changes, and to communicate on a technical level with Engineering partners and other external parties. Anyone, even non-customers, can file issues against Bugzilla, and Red Hat does monitor them and review them for inclusion in errata.

However, Red Hat does not guarantee any SLA for bugs filed directly in Bugzilla (bypassing normal support channels). A review might happen immediately, or after a time span of any length. Issues coming through Support are always prioritized above issues of similar impact and severity filed against Bugzilla. Also, workarounds and hotfixes if possible and appropriate may be provided to customers by Support even before a permanent fix is issued through Red Hat Network.

Red Hat considers issues directly entered into Bugzilla important feedback, and it allows us to provide efficient interaction with the open source development community and as much transparency as possible to customers as issues are processed. Nevertheless, for customers encountering production issues in Red Hat Enterprise Linux, Bugzilla is not the right channel.



---

# About This Course

## Red Hat System Administration II

*Red Hat System Administration II* (RH135) is specifically designed for students who have completed *Red Hat System Administration I* (RH124). *Red Hat System Administration II* focuses on the key tasks needed to become a full time Linux Administrator and to validate those skills via the Red Hat Certified System Administrator (RHCSA) exam. This course goes deeper into enterprise Linux administration, including in-depth coverage of file systems and partitioning, logical volumes, package management and troubleshooting. *Red Hat System Administration II* also focuses on extending the foundation of command-line skills which are invaluable for enterprise level system administration. Students who did not attend *Red Hat System Administration I* but who are prepared to study content at this level are encouraged to attend *RHCSA Fast Track Course* (RH200) instead.

### Objectives

- To complete the training in the skills needed by a full-time Linux system administrator which was begun in *Red Hat System Administration I*
- To prepare students to validate those skills in the RHCSA exam

### Audience and Prerequisites

- Students who have taken *Red Hat System Administration I*

## Structure of the Course

Red Hat training courses are interactive, hands-on, performance-based, real world classes meant to engage your mind and give you an opportunity to use real systems to develop real skills. We encourage students to participate in class and ask questions in order to get the most out of their training sessions.

This course is divided up into a number of *Units* organized around a particular topic area. Each Unit is divided up into multiple *Sections* which focus on a specific skill or task. The unit will start with an introduction to the material, then move on to the first section.

In each section, there will be a *presentation* led by the instructor. During the presentation, it may be a good idea to take notes in your student workbook (this book), and the instructor may remind you to do so. The presentation is followed by a short activity or *assessment* to give you the opportunity to practice with the material or review procedures. After a review of the assessment, the instructor will move on to the next section. At the end of the unit, there will normally be a hands-on lab exercise of some sort (a "criterion test") which will give you an opportunity to learn by doing and review your understanding of the unit's content. Please feel free ask questions in class, or asking the instructor for advice and help during the end-of-unit exercise. We want the classroom environment to be a "low risk" place where you feel comfortable asking questions and learning from things that work and things that do not at first.

## Orientation to the Classroom Network

Two subnets may be used in this course. The primary classroom network is 192.168.0.0/24, and belongs to hosts in the DNS domain "example.com". This network will be used for most classroom activities. Some courses use a second subnet, 192.168.1.0/24, belonging to hosts in the DNS domain "remote.test". This network can be reached from hosts in example.com, and is used in lab exercises which require testing services or security settings from machines (theoretically) outside your administrative control.

Students are each assigned a physical machine (desktopX.example.com on 192.168.0.X) which may host two or more virtual machines for lab activities, serverX.example.com and hostX.example.com.

In some courses, students may also use a non-root account on a test machine in the remote.test domain, remoteX.example.com (192.168.1.X) to test access to network services on their example.com machines in lab activities.

The instructor controls a number of machines which students may see as well. The machine instructor.example.com (also known as instructor.remote.test) is the classroom utility server, providing default routing services, DHCP, DNS name service, one or more YUM repositories of software used by the class, and other network services. It is also connected to the classroom video projector to allow the instructor to display slides and demonstrations. It provides a virtual machine for the instructor, demo.example.com, which the instructor will use for in-class demonstrations.

Machine name	IP addresses	Role
desktopX.example.com	192.168.0.X	Physical student workstation
serverX.example.com	192.168.0.(X+100)	Main student virtual machine
hostX.example.com	192.168.0.(X+200)	Secondary student virtual machine
remoteX.remote.test	192.168.1.X	Student test machine in remote.test domain (shared)
instructor.example.com	192.168.0.254	Physical instructor machine and utility server
instructor.remote.test	192.168.1.254	Identity of instructor.example.com on remote.test network
demo.example.com	192.168.0.250	Instructor virtual demonstration machine

Table 1. Classroom Machines

---

# Internationalization

## Language Support

Red Hat Enterprise Linux 6 officially supports twenty-two languages: English, Assamese, Bengali, Chinese (Simplified), Chinese (Traditional), French, German, Gujarati, Hindi, Italian, Japanese, Kannada, Korean, Malayalam, Marathi, Oriya, Portuguese (Brazilian), Punjabi, Russian, Spanish, Tamil, and Telugu. Support for Maithili, Nepalese, and Sinhala are provided as Technology Previews.

## System-wide Default Language

The operating system's default language is normally set to US English (`en_US.UTF-8`), but this can be changed during or after installation.

To use other languages, you may need to install additional package groups to provide the appropriate fonts, translations, dictionaries, and so forth. By convention, these package groups are always named ***language-support***. These package groups can be selected during installation, or after installation with PackageKit (**System → Administration → Add/Remove Software**) or `yum`.

A system's default language can be changed with **`system-config-language`** (**System → Administration → Language**), which affects the `/etc/sysconfig/i18n` file.

## Per-user Language Selection

Users may prefer to use a different language for their own desktop environment or interactive shells than is set as the system default. This is indicated to the system through the **`LANG`** environment variable.

This may be set automatically for the GNOME desktop environment by selecting a language from the graphical login screen by clicking on the **Language** item at the bottom left corner of the graphical login screen immediately prior to login. The user will be prompted about whether the language selected should be used just for this one login session or as a default for the user from now on. The setting is saved in the user's `~/.dmrc` file by GDM.

If a user wants to make their shell environment use the same **`LANG`** setting as their graphical environment even when they login through a text console or over `ssh`, they can set code similar to the following in their `~/.bashrc` file. This code will set their preferred language if one is saved in `~/.dmrc` or will use the system default if one is not:

```
i=$(grep 'Language=' ${HOME}/.dmrc | sed 's/Language=/\'')
if [ "$i" != "" ]; then
    export LANG=$i
fi
```

Languages with non-ASCII characters may have problems displaying in some environments. Kanji characters, for example, may not display as expected on a virtual console. Individual commands can be made to use another language by setting **LANG** on the command-line:

```
[user@host ~]$ LANG=fr_FR.UTF-8 date
lun. oct. 24 10:37:53 CDT 2011
```

Subsequent commands will revert to using the system's default language for output. The **locale** command can be used to check the current value of **LANG** and other related environment variables.

## Input Methods

IBus (Intelligent Input Bus) can be used to input text in various languages under X if the appropriate language support packages are installed. You can enable IBus with the **im-chooser** command (System → Preferences → Input Method).

## Language Codes Reference

Language	\$LANG value	Language package group
English (US)	en_US.UTF-8	(default)
Assamese	as_IN.UTF-8	assamese-support
Bengali	bn_IN.UTF-8	bengali-support
Chinese (Simplified)	zh_CN.UTF-8	chinese-support
Chinese (Traditional)	zh_TW.UTF-8	chinese-support
French	fr_FR.UTF-8	french-support
German	de_DE.UTF-8	german-support
Gujarati	gu_IN.UTF-8	gujarati-support
Hindi	hi_IN.UTF-8	hindi-support
Italian	it_IT.UTF-8	italian-support
Japanese	ja_JP.UTF-8	japanese-support
Kannada	kn_IN.UTF-8	kannada-support
Korean	ko_KR.UTF-8	korean-support
Malayalam	ml_IN.UTF-8	malayalam-support
Marathi	mr_IN.UTF-8	marathi-support
Oriya	or_IN.UTF-8	oriya-support
Portuguese (Brazilian)	pt_BR.UTF-8	brazilian-support
Punjabi	pa_IN.UTF-8	punjabi-support
Russian	ru_RU.UTF-8	russian-support

Language	\$LANG value	Language package group
Spanish	es_ES.UTF-8	spanish-support
Tamil	ta_IN.UTF-8	tamil-support
Telugu	te_IN.UTF-8	telugu-support
<i>Technology Previews</i>		
Maithili	mai_IN.UTF-8	maithili-support
Nepali	ne_NP.UTF-8	nepali-support
Sinhala	si_LK.UTF-8	sinhala-support

Table 2. Language Codes





## **UNIT ONE**

# **AUTOMATED INSTALLATION OF RED HAT ENTERPRISE LINUX**

## **Introduction**

Topics covered in this unit:

- **system-config-kickstart**
- Serve Kickstart file
- Installation media
- Perform Kickstart installation
- Kickstart file details

## Introductory Overview

Using *Kickstart*, a system administrator can create a single file containing the answers to all the questions typically asked during an installation. This file can then be accessed by the installer to automate installation of Red Hat Enterprise Linux.



### Comparison

Kickstart in Red Hat Enterprise Linux is similar to Jumpstart for Oracle Solaris, or to an unattended installation for Microsoft Windows.

### Basic Steps:

1. Create a Kickstart file	2. Make the Kickstart file available to the installer
3. Boot the installer	4. Point the installer to the Kickstart file

Table1.1. Kickstart Steps



### References

Red Hat Enterprise Linux Installation Guide  
• Chapter 32 - Kickstart Installations

## Create a Kickstart File with system-config-kickstart

1. <b>Create a Kickstart file</b> <ul style="list-style-type: none"><li>• Using <b>system-config-kickstart</b></li><li>• Using a text editor</li></ul>	2. Make the Kickstart file available to the installer
3. Boot the installer	4. Point the installer to the Kickstart file

Table 1.2. Kickstart Steps

The tool **system-config-kickstart** presents a number of dialogs. Each dialog that corresponds to a category of questions normally asked by the interactive installer. Default values are not provided by **system-config-kickstart** for all items. In a Kickstart file, required values which are missing may cause the installer to interactively prompt for an answer or to abort the installation entirely.



### References

- Red Hat Enterprise Linux Installation Guide
  - Chapter 33 - Kickstart Configurator



### Practice Case Study

## Creating a Kickstart File with **system-config-kickstart**

In this exercise, you will pretend that your organization is rolling out Red Hat Enterprise Linux-based workstations for its engineers, and you have been tasked with creating a Kickstart file to facilitate this.

Perform this exercise on your desktopX machine as **student**.

You will install **system-config-kickstart**. When launched, **system-config-kickstart** may ask to install a font; you may safely close this dialog. Use **system-config-kickstart** to create a Kickstart file according to the parameters specified below.

- Choose the appropriate timezone.
- The **root** password should be **redhat**
- The Kickstart should perform a fresh installation from the web server `http://instructor.example.com/pub/rhel6/dvd`
- The MBR should be cleared and the disk initialized.
- The system should have a 100 MB, **ext4** filesystem mounted at **/boot**
- The system should have a 512 MB swap partition
- All remaining disk space should be allocated to an **ext4** partition mounted at **/**
- The **eth0** device should start at boot-time and use DHCP for configuration
- Install the **Base** package set
- Have a post-installation script that adds:

**ENGINEERING WORKSTATION**

to the file **/etc/issue**

- Save the Kickstart file as **/home/student/engineer.cfg**

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*

## Make the Kickstart File Available to Installers

1. Create a Kickstart file	2. <i>Make the Kickstart file available to the installer</i> <ul style="list-style-type: none"><li>• Network servers: FTP, HTTP, NFS</li><li>• DHCP/TFTP server</li><li>• USB disk or CD-ROM</li><li>• Local hard disk</li></ul>
3. Boot the installer	4. Point the installer to the Kickstart file

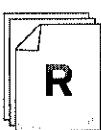
Table 1.3. Kickstart Steps

In order to begin an automated installation, the installer must be able to access the Kickstart file. There are several methods to make the Kickstart file available, and one of the most common is through a network server such as an FTP server, a web server or an NFS server. This method is fairly easy to deploy and makes it easy to manage changes.

Using a DHCP server with TFTP and PXE is more complex to configure and not all locations will allow this method. If you want to configure Kickstart using this method, follow the documentation listed in the references below.

Providing the Kickstart file on USB or CD-ROM can be a convenient way to access the file. Simply place the Kickstart file on the boot media used to start the installation. Be careful when changing the Kickstart, though, as you will need to change it on all of your media.

It can be useful to provide the Kickstart file on the local disk. This allows a quick way to rebuild a development server.



### References

Red Hat Enterprise Linux Installation Guide,

- Section 32.8 (Making the Kickstart File Available)



**Practice Exercise**

## Make the Kickstart File Available to Installers via HTTP

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

1. Login to desktopX as **student**.
2. Become **root** and deploy a web server.



### Note

Recall that you need to Install, Start, Enable and Test

3. Copy **/home/student/engineer.cfg** to **/var/www/html/**
4. Access **http://desktopX/engineer.cfg** from a web browser to test availability.

## Create Boot Media

1. Create a Kickstart file	2. Make the Kickstart file available to the installer
<p>3. <b>Boot the installer</b></p> <ul style="list-style-type: none"><li>• Installation disks</li><li>• PXE</li><li>• boot.iso</li></ul>	4. Point the installer to the Kickstart file

Table 1.4. Kickstart Steps

In older versions of Red Hat Enterprise Linux, the **boot.iso** file could be found in the **images/** directory on the installation media. In Red Hat Enterprise Linux 6 it is a separate download from Red Hat Network.

To create a CD, use **cdrecord boot.iso**.

To create a USB stick, use **dd if=boot.iso of=/dev/sdb1**, however, note that the name **/dev/sdb1** will vary depending on the dynamic assignment of device names.

PXE can be used to start the installation from the network. It is fairly complex to configure and may not be allowed in your location, but PXE can be used to start an installation over the network.



### References

- Red Hat Enterprise Linux Installation Guide
  - Section 2.3 - Making Minimal Boot Media

## Point the Installer to a Kickstart File

1. Create a Kickstart file	2. Make the Kickstart file available to the installer
3. Boot the installer	<p>4. <b><i>Point the installer to the Kickstart file</i></b></p> <ul style="list-style-type: none"> <li>• <code>ks=http://server/dir/file</code></li> <li>• <code>ks=ftp://server/dir/file</code></li> <li>• <code>ks=nfs:server:/dir/file</code></li> <li>• <code>ks=hd:device:/dir/file</code></li> <li>• <code>ks=cdrom:/dir/file</code></li> </ul>

Table1.5. Kickstart Steps

Once you have chosen your Kickstart method, let the installer know where the Kickstart file is located.

For a virtual machine installation, you can provide the Kickstart URL in a box under **URL Options**. For a physical machine, boot using installation media and press the **Tab** key, then enter one of the **ks=** entries above.

In the classroom environment, we will attempt to replicate a physical installation using virtual machines. Create a new virtual machine and choose PXE as the installation method. PXE will provide a boot screen just like booting from installation media. With **Install or upgrade an existing system** highlighted, press the **Tab** key. This will show the boot line as below:

```
> vmlinuz initrd=initrd.img
```

Add a space, then one of the Kickstart entries. The following provide some examples, including the entire boot line. These examples use a Kickstart file named **ks.cfg**, but it could be any name. Once you have entered your Kickstart location, press **Enter** to start the installation.

```
> vmlinuz initrd=initrd.img ks=http://desktopX/ks.cfg
> vmlinuz initrd=initrd.img ks=ftp://desktopX/pub/ks.cfg
> vmlinuz initrd=initrd.img ks=nfs:desktopX/var/ftp/pub/ks.cfg
```

```
> vmlinuz initrd=initrd.img ks=hd:sdb1:/kickstart-files/ks.cfg  
> vmlinuz initrd=initrd.img ks=cdrom:/kickstart-files/ks.cfg
```



## References

### Red Hat Enterprise Linux Installation Guide

- Section 32.8 - Making the Kickstart File Available

### Red Hat Enterprise Linux Installation Guide

- Section 28.4 - Automating the Installation with Kickstart

### Red Hat Enterprise Linux Installation Guide

- Section 28.2 - Kickstart sources



### Practice Exercise

## Initiating a Kickstart Installation

#### ***Before you begin...***

Gracefully shutdown your **serverX (vserver)** virtual machine to reclaim system resources.

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

Use the Kickstart file created earlier (**engineer.cfg**) to deploy a new virtual machine.

1. On desktopX, create a new virtual machine using **virt-manager**. Choose **Network Boot (PXE)** as the installation method. Configure the virtual machine using the defaults if not specified below:
  - Name: engineer
  - Create a disk image on the computer's hard drive: 4 GB
2. Devise and enter an appropriate Kickstart invocation line for the **engineer.cfg** file, then start the installation.

# Modify a Kickstart File

With each installation, the installer, **anaconda**, will create **/root/anaconda-ks.cfg** containing the settings used to generate this system.



## Note

Partitioning information is commented out.

## Example Kickstart file:

```
# Kickstart file automatically generated by anaconda.

#version=RHEL6
install
url --url=ftp://instructor.example.com/pub/rhel6/dvd
lang en_US.UTF-8
keyboard us
network --device eth0 --bootproto dhcp
rootpw --iscrypted $1$UaJvgaTh$KrpFf3K04r9hcZ2hsaa
# Reboot after installation
reboot
firewall --disabled
authconfig --useshadow --enablemd5
selinux --enforcing
timezone --utc America/New_York
bootloader --location=mbr --driveorder=vda --append="crashkernel=auto rhgb quiet"
# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
#clearpart --all --drives=vda

#part /boot --fstype=ext4 --size=100
#part pv.ZS1CDM-iUYu-Gfua-YX0W-MSzd-ftBY-7qTB1E --size=28000
#part swap --size=512
#Volgroup vol0 --pesize=32768 pv.ZS1CDM-iUYu-Gfua-YX0W-MSzd-ftBY-7qTB1E
#logvol /home --fstype=ext4 --name=home --vgname=vol0 --size=500
#logvol / --fstype=ext4 --name=root --vgname=vol0 --size=8192
repo --name="Red Hat Enterprise Linux" --baseurl=ftp://instructor.example.com/pub/rhel6/
dvd/ --cost=100

%packages
@Base
@Console internet tools
@Core
@Desktop
@Desktop Platform
@Development Tools
@General Purpose Desktop
@Graphical Administration Tools
@Internet Browser
@Network file system client
@Printing client
@X Window System
lftp
```

```
mutt  
ntp  
%end  
  
%post  
# Turn on graphical login  
perl -pi -e 's,id:3:initdefault,id:5:initdefault,' /etc/inittab  
%end
```

Why might you want to manually edit a Kickstart file?

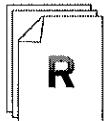
1. The GUI and/or **system-config-kickstart** is unavailable.
2. LVM instructions are needed.
3. Individual packages need to be included or omitted (not just groups).

If you manually edit a Kickstart file, use **ksvalidator** to validate your Kickstart file.

**ksvalidator** is part of the **system-config-kickstart** package, so you may have to install the package first. **ksvalidator** will ensure the keywords are properly used, but it will not validate URL paths, individual packages or groups nor any part of **%post** or **%pre**.

For instance, if you misspell **firewall --disabled** you may get the following output from **ksvalidator**:

```
[student@desktopX]$ ksvalidator /tmp/anaconda-ks.cfg  
The following problem occurred on line 12 of the Kickstart file:  
  
Unknown command: firewall  
  
[student@desktopX]$ ksvalidator /tmp/anaconda-ks.cfg  
The following problem occurred on line 12 of the Kickstart file:  
  
no such option: --dsabled
```



## References

Red Hat Enterprise Linux Installation Guide

- Section 32.4 - Kickstart Options

Red Hat Enterprise Linux Installation Guide

- Section 32.5 - Package Selection

Red Hat Enterprise Linux Installation Guide

- Section 32.6 - Pre-installation Script

Red Hat Enterprise Linux Installation Guide

- Section 32.7 - Post-installation Script



Practice Case Study

## Modify a Kickstart File without system-config-kickstart



### Note

For time's sake you will not perform an installation using this Kickstart file.

As **root** on desktopX, create a copy of **/root/anaconda-ks.cfg** called **/home/student/projman.cfg**. Using only a text editor, modify that file so it meets the following criteria:

1. The installation must be fully automated, and exactly like the current desktopX installation (including partitioning), except...
  - The **Backup Server** package group will be installed
  - The **mtx** package, which is not installed with the **Backup Server** group by default, will be installed
  - None of the existing scripting from **%pre** and **%post** will be used
  - An **/etc/issue** file will be created in **%post**, which reads:

PROJECT MANAGEMENT

2. **ksvalidator** must be able to validate the file

When complete, copy the file to **/var/www/html/**

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*

Test

## Criterion Test

### Performance Checklist

#### Kickstart a Virtual Machine

##### **Before you begin...**

Shutdown the **engineer** virtual machine you created earlier and delete the virtual machine and its disk.

- Boot serverX if it is not running. Copy the `/root/anaconda-ks.cfg` file from serverX to desktopX and call it `/home/student/test.cfg`. Shutdown serverX after you have copied the file to reclaim system resources for the rest of the lab.
- Modify `test.cfg` according to the following criteria:
  - Add `clearpart --all` and `zerombr`, and partition storage according to the following:
    - `/boot` (ext4) 200 MB
    - `swap` 512 MB
    - `/` (ext4) 3 GB
  - Add the `gimp` package
  - Create a `/root/install-date` file with the date and time.
- Copy `test.cfg` to `/var/www/html/` on desktopX. Make sure the file is readable by Apache.
- Start a virtual machine installation using your `test.cfg` Kickstart file. Name the virtual machine `test`. Use PXE as the installation method. and allocate 768 MB of RAM and 1 CPU to the virtual machine. Create a local disk of 4 GB for the virtual disk.
- Reboot your virtual machine when it is finished installing and confirm that it installed correctly.
- After you have successfully completed the exercise and checked your work, remove this virtual machine and its associated storage to clean up.



## Personal Notes



## Unit Summary

### Create a Kickstart File with system-config-kickstart

In this section you learned how to:

- Create a Kickstart configuration from scratch with the **system-config-kickstart** utility
- Identify key configuration elements found inside a Kickstart configuration file

### Make the Kickstart File Available to Installers

In this section you learned how to:

- Make a Kickstart configuration available via the network (NFS, FTP, HTTP) to the installer
- Make a Kickstart configuration available via local media to the installer

### Create Boot Media

In this section you learned how to:

- Create boot media to launch the installer
- Boot over the network to launch the installer

### Point the Installer to a Kickstart File

In this section you learned how to:

- Perform an installation using the now available Kickstart configuration file

### Modify a Kickstart File

In this section you learned how to:

- Modify an existing Kickstart configuration with the **system-config-kickstart** utility or a text editor



## **UNIT TWO**

# **ACCESSING THE COMMAND-LINE**

### **Introduction**

Topics covered in this unit:

- Understanding the bash shell
- Getting to local and remote command lines

## Accessing the Command-line Locally

A *shell* is a program which provides a command line that users can use to interact with the computer. The standard shell program used in Red Hat Enterprise Linux is **bash** (the Bourne-Again Shell, an improved version of the Bourne shell, **sh**, widely used on legacy UNIX systems).

Despite improved graphical tools for configuring Linux systems, there are times when the command line is the most efficient way to complete a task or change the system's configuration. On many Linux servers, graphical interfaces are not installed for various reasons, which is one reason why understanding how to use the shell is important.

### Terminal Windows

The graphical environment in Red Hat Enterprise Linux provides terminal emulation programs that allow you to access a shell through a graphical terminal window. To open a terminal window while logged into a GNOME desktop session, select the following menu item:

**Applications → System Tools → Terminal**

When you open a terminal window, a shell prompt opens immediately as the user that started the graphical terminal program. The shell prompt and the terminal window's title bar will indicate your current user name, host name and working directory.

### Virtual Consoles

Another way to access a shell is from a *virtual console*. A Linux machine's *physical console*, the keyboard and display hardware, supports multiple virtual consoles which act like separate terminals. Each virtual console supports an independent login session.

If the graphical environment is available, it will run on the *first* virtual console in Red Hat Enterprise Linux 6. Five additional text login prompts are available on consoles two through six (or one through five if the graphical environment is turned off). With a graphical environment running, access a text login prompt on a virtual console by holding **Ctrl+Alt** and pressing a function key (**F2** through **F6**). Press **Ctrl+Alt+F1** to return to the first virtual console and the graphical desktop.



#### Note

In Red Hat Enterprise Linux 5 and earlier, the first six virtual consoles always provided text login prompts. The graphical environment, when available, ran on virtual console seven (accessed through **Ctrl+Alt+F7**).

## Shell Basics

Here are a few tips on how to use the shell once you have logged in:

- By convention, a \$ shell prompt shows you are logged in as a regular user, while a # shell prompt shows you are logged in as the root superuser.
- To switch to the superuser, type **su -** and enter the root password when prompted. When you exit the root shell with **logout** or **Ctrl+D**, you will be returned to the shell from which you ran **su -**.
- After log in, you can type a few commands to get information about your shell session: **pwd** (your current directory), **id** (the user and group you are logged in as), and **tty** (the current terminal device).
- In *Red Hat System Administration I*, we discussed a number of commands used for working with files from the command line. Use **cd** to go to other directories, **cp** to copy files, **mv** to move and rename files, **mkdir** to create directories, **rm** to remove files and **rmdir** to remove empty directories.
- Remember that the local system manual is a useful resource. Type **man** followed by any of the commands above to learn more about how they work.

### Workshop



## Accessing the Command-line Locally

Use virtual consoles to access a shell interface.

*Follow along with the instructor as you complete the steps below.*

1. From your desktop, switch to a virtual console by pressing **Ctrl+Alt+F2** and log in
2. Switch to a different virtual console by pressing **Ctrl+Alt+F3**
3. Switch back to the other virtual console by pressing **Ctrl+Alt+F2**
4. Escalate privileges with **su -**
5. Leave the privileged shell by typing **exit**
6. To log out completely (returning to the login prompt) type **exit** again
7. Return to your first virtual console (usually your desktop) by pressing **Ctrl+Alt+F1**



### References

**bash(1)**, **su(1)**, **Xorg(1)** (see "KEYBOARD"), **pts(4)**, and **console(4)** man pages

*Note: some details of console(4) involving init(8) and inittab(5) are outdated.*



**Practice Quiz**

**Local Command-line Access**

1. Type \_\_\_\_\_ to switch to the second virtual terminal.
2. Type \_\_\_\_\_ to switch to the root account.
3. Type \_\_\_\_\_ to logout of an **su** – shell
4. Type \_\_\_\_\_ to logout of the virtual terminal.
5. Type \_\_\_\_\_ to return to the GUI.

# Accessing the Command-line Using ssh

The Secure Shell, **ssh**, is used to securely run a shell on a remote system. If you have a user account on a remote Linux system providing SSH service, **ssh** is the command normally used to remotely log in to that system. The **ssh** command can also be used to run an individual command on a remote system.

Here are some examples of **ssh** command syntax for remote login and remote execution:

- Create a remote interactive shell as current user, then return when you are done (**exit**)

```
[student@host ~]$ ssh remotehost
student@remotehost's password:
[student@remotehost ~]$ exit
Connection to remotehost closed.
[student@host ~]$
```

- Connect to a remote shell as a different user (**remoteuser**) on a selected host (**remotehost**):

```
[student@host ~]$ ssh remoteuser@remotehost
remoteuser@remotehost's password:
remoteuser@remotehost ~]$
```

- Execute a single command (**hostname**) on a remote host (**remotehost**) and as a remote user (**remoteuser**) in a way that returns the output to the local display:

```
[student@host ~]$ ssh remotehost hostname
student@remotehost's password:
remotehost.example.com
[student@host ~]$
```

```
[student@host ~]$ ssh remoteuser@remotehost hostname
remoteuser@remotehost's password:
remotehost.example.com
[student@host ~]$
```

The **w** command displays a list of users currently logged in to the computer. This is especially useful to show which users are logged in using **ssh** from which remote locations, and what they are doing.

```
[student@host ~]$ w
USER   TTY      FROM          LOGIN@    IDLE   JCPU   PCPU WHAT
student  tty1     :0           Wed08    2days  1:52m  0.07s pam: gdm-passwo
root    tty6     -           12:33    4:14m  16.27s 15.74s -bash
student  pts/0     :0.0        Wed08    5:11    1.63s  1.60s /usr/bin/gnome-
student  pts/1     :0.0        Wed08    43:44   14.48s 13.81s vim hello.c
student  pts/3     :0.0        Wed14    0.00s  0.06s  0.06s w
visitor  pts/6     server2.example. 09:22    3:14    0.02s  0.02s -bash
```

In the example above, user **student** logged in to virtual console 1 (**tty1**) through the graphical login (:0) at about 08:00 on Wednesday. User **student** currently has three pseudoterminals open (**pts/0**, **pts/1**, and **pts/3**) started by the graphical environment; these are almost certainly terminal windows. In one of them, **student** is editing **hello.c**. User **root** is logged in

to virtual console 6, starting at 12:33 today. User *visitor* logged in at 09:22 today from the host server2.example.com (note that the name has been truncated), probably using **ssh**, and has been sitting idle at a shell prompt for three minutes and fourteen seconds.

## SSH Host Keys

SSH secures communication through public-key encryption. When an **ssh** client connects to an SSH server, before the client logs in, the server sends it a copy of its *public key*. This is used to set up the secure encryption for the communication channel and to authenticate the server to the client.

The first time a user uses **ssh** to connect to a particular server, the **ssh** command stores the server's public key in the user's `~/.ssh/known_hosts` file. Then, every time the user connects after that, it makes sure that it gets the same public key from the server by comparing the server's entry in the `~/.ssh/known_hosts` file to the public key the server sent. If the keys do *not* match, the client assumes that the network traffic is being hijacked or that the server has been compromised and breaks the connection.

This means that if a server's public key is changed (because the key was lost due to hard drive failure, or replaced for some legitimate reason), users will need to update their `~/.ssh/known_hosts` files to remove the old entry in order to log in.

- Host IDs are stored in `~/.ssh/known_hosts` on your local system:

```
$ cat ~/.ssh/known_hosts
remotehost,192.168.0.101 ssh-rsa AAAAB3Nz...
```

- Host keys are stored in `/etc/ssh/ssh_host_key*`

```
$ ls /etc/ssh/*key*
ssh_host_dsa_key      ssh_host_key          ssh_host_rsa_key
ssh_host_dsa_key.pub  ssh_host_key.pub      ssh_host_rsa_key.pub
```



### Note

An even better approach is to add entries matching a server's `ssh_host_*key.pub` files to user `~/.ssh/known_hosts` or the system-wide `/etc/ssh/ssh_known_hosts` in advance when the public keys change. See **ssh-copy-id(1)** for an advanced way to manage ssh keys.



### References

**ssh(1)**, **w(1)**, and **hostname(1)** man pages



Practice Performance Checklist

## Access Remote Command-line

- Login as student on your desktopX machine.
- ssh** to your serverX machine. Accept the host key if asked.
- Run the **w** command and **exit**.
- ssh** to your serverX machine as **root**.
- Run the **w** command and exit.
- Remove the ssh known hosts file.
- ssh** to serverX as **root** again. Accept the ssh key, login and then exit the session.
- Use **ssh** non-interactively to run the **hostname** command on serverX as **root**.

Test

## Criterion Test

Exercise

### Using the command-line

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

1. Switch to the second virtual terminal and login as **student** on your desktopX machine.
2. Become **root**.
3. **ssh** to your serverX machine. Accept the host key if asked.
4. Run the **w** command and **exit**.
5. **ssh** to your serverX machine as **student**.
6. Run the **w** command and **exit**.
7. Remove the **ssh** known hosts file.
8. **ssh** to serverX as **student** again and note that you are prompted for the host's key again. Accept the key.
9. Once logged in, **exit** back to your shell on desktopX.
10. Use **ssh** non-interactively to run the **ip route** command on serverX as **root**.



## Personal Notes



## Unit Summary

### Accessing the Command-line Locally

In this section you learned how to:

- Access the command line locally with terminal windows
- Access the command line locally with virtual consoles

### Accessing the Command-line Using ssh

In this section you learned how to:

- Access the command line remotely using **ssh**
- Identify host IDs and keys storage locations



## **UNIT THREE**

# **INTERMEDIATE COMMAND-LINE TOOLS**

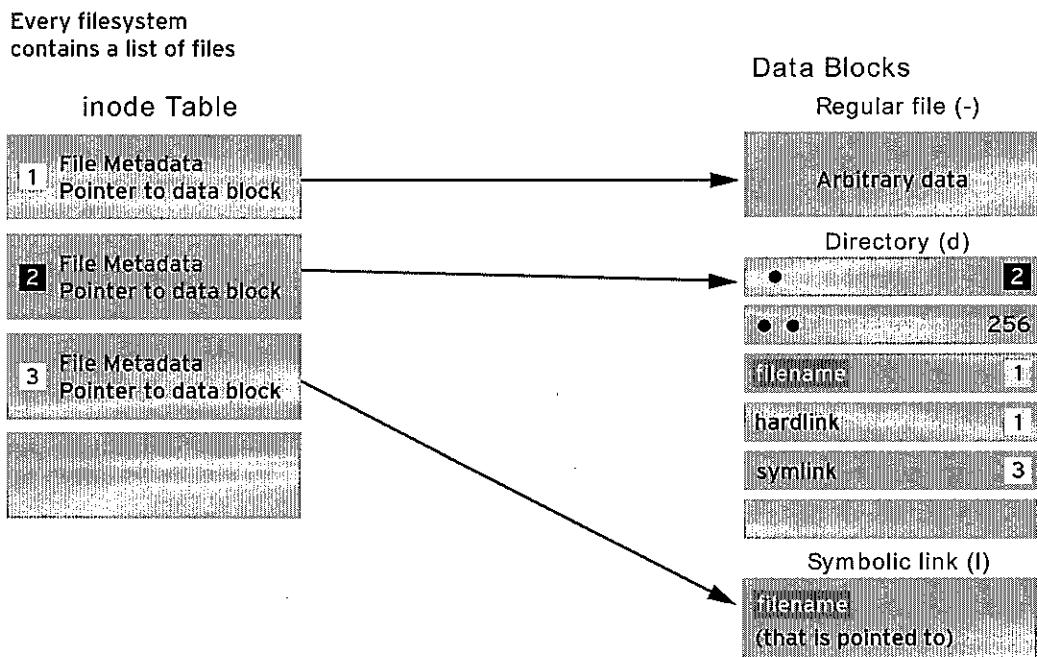
## **Introduction**

Topics covered in this unit:

- Creating and using hard links
- Archiving and compressing files
- Using the `vim` text editor

## Using Hard Links

In *Red Hat System Administration I*, you were introduced to *symbolic links*, also known as *symlinks* or *soft links*. A symlink is a special file that points to another file by name, and which can be treated as if it were that file, within certain limits. In this section, you will learn about another kind of link, called a *hard link*, and how they differ from symlinks.



Every file on a POSIX file system such as ext4 has an *index node*, or *inode*, associated with it. The inode contains the basic information about the file: its permissions, timestamps, and pointers to the data stored in the file. When you create a new file, an unused inode on the file system is allocated to it, and an entry is added to the directory the new file is in, mapping the name of the file to the number of the inode. A hard link allows you to set an additional name pointing to the same inode number from somewhere else on the same file system.

Since hard links point a name to inode data, they can not be used to associate two files on separate file systems. Soft links, which point a name to another name, can cross file systems and even point to directories. On the other hand, all hard links equally point to the same data; no name is more important than any other, and any can be removed independently. With soft links, removing the original file also breaks all soft links pointing to it.

Hard links are created by the **ln** command, just like symlinks, but without the **-s** option:

```
[user@host ~]$ ln original hardlink
```

Hard Links	Symbolic (or Soft) Links
Multiple names pointing to same inode	Additional names pointing to original name
Increments link count	Separate file
All names are equal	Additional names can be broken
Data preserved until all names removed	Data lost if original name removed
Must be on same file system	Can span across file systems

Table 3.1. Comparison between types of links

## Creating and Viewing Links

1. Create a symbolic link:

```
$ touch original
$ ln -s original symlink
```

2. Create a hard link:

```
$ ln original hardlink
```

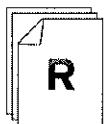
3. View the inode, link count, and symbolic link reference:

```
$ ls -il original symlink hardlink
65661 ① -rw-r--r--, 2 ② root root 0 Jan 26 12:20 hardlink
65661 -rw-r--r--, 2 ③ root root 0 Jan 26 12:20 original
65662 lrwxrwxrwx. 1 ④ root root 8 ⑤ Jan 26 12:20 symlink⑥ -> original⑦
```

- ➊ The first column of output from **ls -il** is the inode number.
- ➋ The link count. This increments when you create a new hard link, and decrements when you remove one of the hard links.
- ➌ This is the size in bytes of the file (the name "**original**" is eight one-byte characters).
- ➍ The name of the symbolic link.
- ➎ The name of the file to which the symbolic link points.

The files **original** and **hardlink** have the same inode number (65661) because there is really only one file with multiple names associated with it. Removing either file name would leave the other file name intact, but would reduce the link count (from 2 to 1).

The symbolic link (**symlink**) has a *different* inode number, because it is a different file. Your ability to access the **symlink** file is based on the permissions of **original** (rw-r-r-). The rwxrwxrwx permissions shown on **symlink** are meaningless. Remove **original**, and **symlink** points to a missing file (it is a *broken link*).



## References

**ln(1)**, **ls(1)**, and **symlink(7)** man pages



Practice Quiz

## Comparing Hard and Symbolic Links

1. A \_\_\_\_\_ can be used across file systems.
2. A \_\_\_\_\_ increases the link count.
3. When you create a \_\_\_\_\_ you can remove the original file without losing data.



#### Practice Exercise

## Implement Hard and Symbolic Links

Use this task to practice creating and examining hard and symbolic links.

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

1. Login to the desktopX machine as student.
2. Create a file named **test1** in your home directory.
3. Create a symbolic link in your home directory named **symlink** pointing to the **test1** file and check that **symlink** points to the contents of **test1**.
4. Create a hard link in your home directory named **hardlink** pointing to the **test1** file.
5. What is the link count on the **test1** file?
6. Remove the **test1** file.
7. What happened to **symlink**?
8. What is the link count on **hardlink**?
9. How would you get your data back to the **test1** file?

# Archives and Compression

Archiving and compressing files are useful when creating backups and transferring data across a network. One of the oldest and most common commands for creating and working with backup archives is the **tar** command (which originally stood for tape archiver).

With **tar**, you can gather large sets of files into a single file (archive). You can indicate that the archive should be compressed using **gzip** or **bzip2** compression. Some options let you back up special features associated with your backed up files, such as extended file attributes and SELinux contexts.

The **tar** command also can list the contents of archives or extract their files to your current system. Examples of how to use the **tar** command are included in this section.

## Key tar Options

1. \_\_\_\_\_ = create
2. \_\_\_\_\_ = extract
3. \_\_\_\_\_ = test
4. \_\_\_\_\_ = verbose
5. \_\_\_\_\_ = file name
6. \_\_\_\_\_ = gzip
7. \_\_\_\_\_ = bzip2

To use the **tar** command, one of the three following options is required **c** (create an archive), **x** (extract an archive) or **t** (test or list the contents of an archive). Other options let you add verbosity (**v**), indicate the name of the archive file to create or extract (**f filename**) and set the type of compression to use (**g** for gzip or **j** or **bzip2**).

## Example tar Syntax

Create (**c**) a gzip-compressed (**z**) archive file (**f /tmp/etc.tar.gz**) of the **/etc** directory. Be verbose (**v**) with the output. The **tar** command has no special ability to read files for which you don't have read access, so run this command as root to backup the entire **/etc** directory.

```
# tar cvzf /tmp/etc.tar.gz /etc
```

Show a long (**v**) listing (**t**) of files in a bzip2-compressed (**j**) archive file (**f /tmp/abc.tar.bz2**).

```
$ tar tvjf /tmp/abc.tar.bz2
```

Extract (**x**) and view (**v**) all files from a gzip-compressed (**z**) archive (**f /tmp/etc.tar.gz**) to the current directory.

```
$ tar xvzf /tmp/etc.tgz
```

Additionally, **gzip** and **bzip2** can be used independently to compress files. **gunzip** and **bunzip2** are the corresponding decompress commands. For example, **bunzip abc.tar.bz2** results in the uncompressed tar file **abc.tar**. Likewise, **gunzip /tmp/etc.tgz** results in the uncompressed tar file **etc.tar**.

Use this space for notes

---



## References

**tar(1)**, **bzip2(1)**, and **gzip(1)** man pages



Practice Exercise

## Archive and Compress Files

Backup the **/etc** directory from serverX to a compressed tar archive file. Use the **rsync** command to copy that archive to your desktopX system. Then extract the files to the **/backups** directory on desktopX. Create the **/backups** directory if it doesn't already exist.

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

1. Login to serverX as **root**.
2. Create an archive of **/etc** using **gzip** compression. Save the file as **/tmp/etc.tar.gz**.
3. Copy the **/tmp/etc.tar.gz** file from your serverX to the **/backups** directory on your desktopX machine.
4. Extract the compressed archive to **/backups** on desktopX.

## Introduction to vim

VIM (vi Improved) is a powerful text editor which is more sophisticated than **gedit**. Also, unlike **gedit**, you can always expect **vim** (or its predecessor, **vi**) to be available in any Linux system.

The **vim** editor supports sophisticated text manipulations which are very useful for system administration. In fact, if you are familiar with the **vi** utility, you will find that **vim** includes **vi** features, plus many other features such as split screen editing, color formatting of known file types, and text highlighting for copying and changing text, to name a few.

One reason why **vim** is powerful but tricky to learn is that it is a *modal* editor; it works differently depending on which mode it is in. There are three modes you need to know to start with: normal mode (also called *command mode*), *insert mode*, and *ex mode*.

### Primary vim Modes

1. \_\_\_\_\_ mode is used for file navigation, cut and paste, and simple commands.
2. \_\_\_\_\_ mode is used for normal text editing
3. \_\_\_\_\_ mode is used to save, quit, search and replace, and perform other complex operations.

### Command Mode

After opening the **vim** editor, you begin in command mode. This is the basic mode you return to when done with an edit and preparing to save, search and replace text, or perform other operations. In this mode, you can move around the file (using **PageUp**, **PageDown**, or arrow keys), and typing characters can execute many different commands to do such things as selecting, copying, or pasting text. One useful command here is **u**, to undo the last change. If you are confused about what mode **vim** is in, you can type **Esc** a couple of times to return to command mode.

### Insert Mode

From command mode, press the **i** key to go into insert mode at the cursor's current position. Then you can begin entering text. You can use the arrow keys to move the cursor in **vim** while remaining in insert mode. Press the **Esc** key to leave insert and return to command mode.

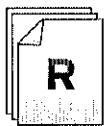
### ex Mode

From command mode, press a **:** character to go into ex mode and the cursor moves to the bottom of the screen. Press **:q** (to quit), **:w** (to write changes to the file), or **:wq** to write and quit. Use the **!** character to override a default action. For example **:q!** lets you quit without saving and **:w!** lets you overwrite a read-only file (if you own it). Other commands are available in ex mode, including an on-line help system under **:help**.



## Important

In Red Hat Enterprise Linux, **vi** runs **vim** in a simpler vi-compatibility mode without all the enhancements of **vim**. This can be confusing, because **vi** is an alias for the full-featured **vim** command for all users with UIDs above 200. Therefore, if you run **vi** as a typical user, you will get **vim**; but if you run **vi** as root, you will get the simplified **vi** command. Root can work around this by running **vim** explicitly.



## References

**vim(1)** man page

## Basic vim Workflow

The instructor will demonstrate a typical example of how to edit a file using basic **vim** commands. Much more sophisticated things are possible, but this basic workflow is the minimum needed to be productive with **vim**.

### vim Operations

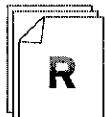
1. Open a file: **vim filename**
2. Use arrow keys to position the cursor.
3. Switch to insert mode (from command mode): **i**
4. Enter your text.
5. Return to command mode: **Esc**
6. Switch to ex mode (from command mode): **:** (press the colon key)
7. Save the file (from ex mode): **w**  
*or*  
Save and Quit (from ex mode): **wq**  
*or*  
Quit, abandoning changes (from ex mode): **q!**

Also, remember that in command mode you can undo changes with the **u** command.



### Note

The **vimtutor** command, included with **vim**, runs a tutorial that can help you learn more about **vim** outside of class. Note that **vimtutor** is available in multiple languages and automatically respects the current **\$LANG** setting; see the Introduction to this book and the **vimtutor(1)** man page for more details.



### References

**vim(1)** and **vimtutor(1)** man pages



### Practice Exercise

## Edit Files with vim

Create and edit a small text file to practice some **vim** commands. Then edit the **/etc/issue** file and see how the content of that file shows up in the login prompts of your virtual consoles.

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

1. Open a shell on your desktopX machine and become **root**
2. Open a practice file: **vim /tmp/testfile.txt**
3. Press **i** to go into insert mode. Then type in text so it appears as follows:

```
Change the last word in this line to goose: duck
Quack Quack Remove the first two Quack words.
Copy this line so it appears twice
Make sure this is the first line in the file.
```

4. Press the **Esc** key and type **1G** to return to the first line of the file.
5. On the first line, change the word **duck** to **goose**.
6. On the second line, delete the first two occurrences of Quack.
7. Copy the third line, then paste it in so it appears twice.
8. Delete the final line, then paste it so it appears as the first line in the file.

When you are done, the whole text should appear as follows:

```
Make sure this is the first line in the file.
Change the last word in this line to goose: goose
Remove the first two Quack words.
Copy this line so it appears twice
Copy this line so it appears twice
```

9. Save and quit the file (type **:wq**).
10. Next, use vim to edit **/etc/issue** and add the text

```
Welcome!
```

to the top of the file.

11. Test by switching to a virtual console (**Ctrl+AltF2**) and pressing **Enter** to refresh the prompt. The text **Welcome!** should appear at the top of your login prompt.

Test

## Criterion Test

Exercise

### Practice using ln, tar and vim

Combine the skills you learned in this unit by creating links (hard and soft) with the **ln** command, creating a **tar** archive file, and using the **vim** text editor.

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

1. Login to serverX as **student**
2. Create a symlink to **/etc/passwd** in your home directory.
3. Create a new file in your home directory named **newfile**.
4. Create a hardlink to **newfile** named **newhardlink**.
5. Archive and compress (using bzip2 compression) your home directory to **/tmp/student-home.tar.bz2**.
6. Create a new file in your home directory named **myvimfile.txt** using **vim**, and include the following line:

I wrote this using vim!



## Personal Notes



## Unit Summary

### Using Hard Links

In this section you learned how to:

- Identify differences between hard and symbolic links
- Create and use hard links

### Archives and Compression

In this section you learned how to:

- Archive and compress files using command-line tools (**tar**, **gzip**, **bzip2**)

### Introduction to vim

In this section you learned how to:

- Identify the three primary modes of **vim**
- Use some basic commands with the **vim** editor

### Basic vim Workflow

In this section you learned how to:

- Create and modify files with **vim**



## **UNIT FOUR**

# **REGULAR EXPRESSIONS, PIPELINES AND I/O REDIRECTION**

## **Introduction**

Topics covered in this unit:

- Regex pattern matchings
- grep text filter
- Redirecting input/output

# Basic Regular Expressions

*Regular expressions* are special text strings that are used to search for and match patterns in text. They provide a concise but powerful way to express exactly the text that a search should match. They are not as user-friendly to a novice as a graphical search tool might be, but they allow complex conditions to be expressed precisely with some practice. Regular expressions are used by many Linux command line tools, including **less**, **man**, **vim**, and others we have not yet introduced.

Regular expressions are sometimes called *regexp*s or *regexes* for short.

The basic rule of regular expressions is that normal characters that appear in a regex string are treated as the characters we want to match, for the most part. Punctuation or special characters are used as wildcards, to indicate repetition, to mark line or word boundaries, or for other special matching purposes. Regular expressions are also *greedy*; they match on the longest possible string that fits.

The table below provides some examples of regular expression syntax.

Expr	Definition	Regex example	Example matches
d	Literal: the letter "d"	dog	<b>dog</b> <b>dogma</b> <b>slumdogs</b>
*	Modifier: zero or more of the previous character	hel*o	<b>hello</b> <b>Theophilus</b> <b>helllllllo</b>
.	Wildcard: any single character	test.txt	<b>test.txt</b> <b>test0txt.jpg</b> <b>mytest!txt</b>
[]	Wildcard: any single character in the set	file[1234]	<b>file1</b> <b>file2</b> <b>file3.txt</b> <b>somefile4</b>
[^ ]	Wildcard: any single character <i>not</i> in the set	file[^0123456789]	<b>filea</b> <b>fileA</b> <b>fileb.txt</b> <b>somefile%</b>
^	Anchor: line begins with...	^Test	<b>[line begins]Test</b>

Expr	Definition	Regex example	Example matches
			<i>[line begins]Testing. 1, 2, 3, 4.</i>
\$	Anchor: line ends with...	test\$	<b>test</b> <i>[end of line]</i> <b>some test</b> <i>[end of line]</i>
.*	combination of . (any character) plus * (zero or more)	^Test.*123	<i>[line begins]Testing1234</i> <i>[line begins]Test123.txt</i> <i>[line begins]Test this has 123456.</i>
\	treat the next character as literal	test\.S	<b>test.</b> <i>[end of line]</i>

Table 4.1. Regular Expression Syntax



### Note

There are two variations of regular expression syntax, *basic* and *extended*, as documented in **regex(7)**. They differ in some minor details which do not affect the discussion above.

## More Regular Expression Examples

1. What does **h[aiou]t** match?

2. What does **b[ol][oa]t** match?



### Note

*Advanced students.* Using ranges such as **[a-z]** to match such things as all lower case characters is error prone since sort order varies depending on the setting of **\$LANG** when the regex is run. Using POSIX character classes such as **[:lower:]** is generally better; see the **regex(7)** man page for details. We will not discuss this further in this course.



### References

[regex\(7\) man page](#)



### Practice Quiz

## Matching Regular Expressions

For each of the following choose all the items that match the regular expression.

1. **ba[nr]k**

*(select one or more of the following...)*

- a. bank
- b. banrk
- c. bark
- d. bakk

2. **^root**

*(select one or more of the following...)*

- a. root:x:0:0:root:/root:/bin/bash
- b. operator:x:11:0:operator:/root:/sbin/nologin
- c. root is the best!
- d. You are not root.

3. **root**

*(select one or more of the following...)*

- a. root:x:0:0:root:/root:/bin/bash
- b. operator:x:11:0:operator:/root:/sbin/nologin
- c. root is the best!
- d. You are not root.

4. **r..t**

*(select one or more of the following...)*

- a. root:x:0:0:root:/root:/bin/bash
- b. operator:x:11:0:operator:/root:/sbin/nologin
- c. root is the best!
- d. You are not root.



Practice Quiz

## Create a Regular Expression

For each of the following requirements, devise a regular expression that would match.

1. Line begins with "Test" or "test"

---

2. Line ends with "end."

---

3. The entire line is:

This is a test.

---

4. Any of the following names:

file5 file6 file7 file8

---

5. Any of the following names:

file2 file4 file6 file8

---



**Practice Exercise**

## Regular Expressions in man Pages

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

Open the **passwd(1)** man page and perform the following regular expression searches. Recall that you use the forward slash, /, to search man pages.



### Important

Be sure to return to the top of the man page in between each of these searches by pressing the **g** key.

1. Search for **exit**
2. Search for a line that begins with **exit**
3. Search for **password**
4. Search for **password.**
5. Search for anything up to (and including) **pass**

## Using grep

The **grep** command is a **General Regular Expression Parser**; it searches a file for strings matching a given regular expression, and by default it then prints out any line containing a string that matches.

There are many options which can be set for **grep** which affect its output. Some of the most useful include **--color** to colorize the pattern on the line that matched, **-i** which matches the regex in a case insensitive way, and **-v** which prints out any line that *does not* contain a string that matches the regex.

The **grep** command can be a surprisingly useful tool when analyzing configuration files or searching for information stored in text files on the system.

We will explore the use of **grep** by extracting lines from the **/etc/passwd** file. Take notes about the pattern that each of these commands matches.

- **grep 'root' /etc/passwd**
- **grep --color 'root' /etc/passwd**
- **grep '^root' /etc/passwd**
- **grep ':/bin/bash\$' /etc/passwd**
- **grep ':/home/.\*:/' /etc/passwd**

Use this space for notes

---



### References

**grep(1), passwd(5) man pages**



Practice Quiz

## Create grep Commands

Devise **grep** commands to fulfill the requirements listed below based on the **/etc/passwd** file.

1. Print all usernames that begin with the letter r.

---

2. Print all usernames that begin with the letter g.

---

3. Print all accounts whose shells (last column) are **/sbin/nologin**

---

4. Print all accounts that have a UID or GID (third or fourth columns) of 0.

---

5. Print all accounts that have a UID or GID in the range of 10-19.

---

# Pipelines and Redirection

Pipelines and I/O redirection are two of the most powerful features of the command line in Linux.

*I/O redirection* allows you to redirect the standard output or error messages from a program into a file, to save it or analyze it later, or to suppress it from appearing on the terminal. You can also read input from a file, rather than the keyboard, into a command line program.

*Pipes* allow you to connect the standard output messages from a program into the input of another program. This allows multiple smaller programs to be chained together into a *pipeline*, each program acting on the output of the one before.

Name	Description	Number	Default
STDIN	Standard input	0	Keyboard
STDOUT	Standard output	1	Terminal
STDERR	Standard error	2	Terminal

Table 4.2. Channels

Keyword	Definition	Examples
>	Redirect STDOUT to a file (overwrite)	<code>date &gt; /tmp/file</code> <code>date &gt; /tmp/file</code> <code>grep '^sbin/nologin\$' /etc/passwd &gt; /tmp/file2</code> <code>cat file1 file2 file3 file4 &gt; /tmp/file</code>
>>	Redirect STDOUT to a file (append)	<code>date &gt;&gt; /tmp/file</code> <code>date &gt; /tmp/file</code> <code>ls &gt;&gt; /tmp/file</code> <code>find /etc -name passwd &gt;&gt; /tmp/file</code>
2>	Redirect STDERR to a file (overwrite)	<code>find /etc -name passwd 2&gt; /tmp/errors</code> <code>find /etc -name passwd &gt; /tmp/output 2&gt; /tmp/errors</code>
2>/dev/null	Discard errors by redirecting to /dev/null	<code>find /etc -name passwd 2&gt; /dev/null</code> <code>find /etc -name passwd &gt; /tmp/output 2&gt; /dev/null</code>
2>&1	Combine STDERR with STDOUT	<code>find /etc -name passwd &gt; /tmp/all 2&gt;&amp;1</code> <code>find /etc -name passwd 2&gt;&amp;1   less</code>
<	Redirect STDIN	<code>grep 'root' &lt; /etc/passwd</code>
	Pipe-send STDOUT from one command to the STDIN of another command.	<code>ls /usr/lib64   grep '^lib'</code> <code>grep '^sbin/nologin\$' /etc/passwd   lpr</code> <code>sort /etc/passwd   mail -s "Sorted Accounts" root@demo.example.com</code> <code>ls /usr/lib64   less</code>

Table 4.3. Redirection Operators



## Important

**2>&1** redirects standard error to the same place as standard output. If you redirect standard output on the same command line, **2>&1** must be *last*:

```
[user@host ~]$ find / -user student > output 2>&1
```



## Warning

**program > file** or **program 2> file** commands will overwrite the contents of **file** if **file** already exists! Be very careful when using output or error redirection.



## References

**bash(1)**, **cat(1)**, **sort(1)**, **grep(1)**, **lpr(1)**, **less(1)**, **mail(1)**, **find(1)** man pages



## Practice Quiz

## Pipelines and Redirection

Devise commands that meet the criteria listed below

1. List all files in **/usr/share/doc** that end with the number four.

---

2. Print all lines in **/etc/hosts** that have a number in them

---

3. Print the line in **/etc/hosts** that has a **127.0.0.1**

---

4. Run the following command as **student**, and redirect STDOUT to **/tmp/output.txt** and redirect STDERR to **/tmp/error.txt**:

```
find /etc -name "host"
```

---

5. Run the following command as **student**, and redirect both STDOUT and STDERR to the **/tmp/all.txt** file.

```
find /etc -name "host"
```

---

6. Sort the **/etc/passwd** file and send it to the default printer

---

7. Print out lines in **/etc/passwd** that have a three digit number between colons (:).



Test

## Criterion Test

Exercise

### Regular expressions, pipelines and I/O redirection

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

1. Login to serverX as **student**
2. Print all lines in **/etc/hosts** that do not begin with a hash mark (#). [HINT: search for **Invert** in the **grep(1)** man page]. Redirect the output to **/tmp/hosts**.
3. Append the output of **date** and **hostname** to the **/tmp/hosts** file.
4. Edit **/tmp/hosts** with **gedit** or **vim** and change **127.0.0.1** to **127.0.0.2**. Comment the **date** and **hostname** output by adding a hash (#) to the beginning of the line.
5. Using the **lpr(1)** man page, print out the lines that have the word **file** followed by zero or more of the letter **s**. Redirect that output to the **/tmp/lpr.man** file



## Personal Notes



## Unit Summary

### Basic Regular Expressions

In this section you learned how to:

- Identify basic regular expression syntax using literals, modifiers, wildcards and anchors

### Using grep

In this section you learned how to:

- Use command-line tools to search through files and output

### Pipelines and Redirection

In this section you learned how to:

- Redirect command output
- Use pipelines to manage output





## **UNIT FIVE**

# **NETWORK CONFIGURATION AND TROUBLESHOOTING**

## **Introduction**

Topics covered in this unit:

- Network Configuration Files
- Basic Troubleshooting Process
- Network Troubleshooting Toolkit

# Understanding Network Configuration Files

## Network Interface Names

The Linux kernel names interfaces with a specific prefix depending on the type of interface. For example, all Ethernet interfaces start with **eth**, regardless of the specific hardware vendor. Following the prefix, each interface is numbered, starting at zero. For example, **eth0**, **eth1**, and **eth2** would refer to the first, second, and third Ethernet interfaces. Other interface names include **wlan0** for the first wireless device, **virbr0** for the internal bridge set up for virtual hosts, **bond0** for the first bonded network device, and so on.

## Network Interface Configuration

**/sbin/ip** is used to show or temporarily modify devices, routing, policy routing, and tunnels.

```
[root@demo ~]# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:00:fa brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.250/24 brd 192.168.0.255 scope global eth0
        inet6 fe80::5054:ff:fe00:fa/64 scope link
            valid_lft forever preferred_lft forever
[root@demo ~]# ip -s link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:00:fa brd ff:ff:ff:ff:ff:ff
    RX: bytes packets errors dropped overrun mcast
        91449      520      0      0      0      0
    TX: bytes packets errors dropped carrier collsns
        14020       99      0      0      0      0
[root@demo ~]# ip route
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.250 metric 1
default via 192.168.0.254 dev eth0 proto static
```

### Note

**ip -6 route** shows the IPv6 routing table.



## Hostname Resolution

The **hostname** command displays or temporarily modifies the system's fully-qualified hostname.

```
[root@demo ~]# hostname
demo.example.com
```

The **stub resolver** is used to convert hostnames to IP addresses or the reverse. The contents of the file **/etc/hosts** is checked first.

```
[root@demo ~]# cat /etc/hosts
192.168.0.250   demo.example.com          demo    # Added by NetworkManager
127.0.0.1       localhost.localdomain     localhost
::1             demo.example.com           demo    localhost6.localdomain6 localhost6
```

If an entry is not found in that file the stub resolver looks for the information from a DNS nameserver. The **/etc/resolv.conf** file controls how this query is done:

- **nameserver**: the IP address of a nameserver to query. Up to three nameserver directives may be given to provide backups if one is down.
- **search**: a list of domain names to try with a short hostname. Both this and **domain** should not be set in the same file; if they are, the last instance wins. See **resolv.conf(5)** for details.

```
[root@demo ~]# cat /etc/resolv.conf
# Generated by NetworkManager
domain example.com
search example.com
nameserver 192.168.0.254
```

The **getent hosts *hostname*** command can be used to test hostname resolution.

## Modifying Network Configuration

**NetworkManager** may be installed on Red Hat Enterprise Linux 6. It consists of a core daemon, a GNOME Notification Area applet that provides network status information, and graphical configuration tools that can create, edit and remove connections and interfaces.

To change a NetworkManager-managed eth0 interface from using DHCP to using a static IP address:

1. Right-click the NetworkManager icon in the top Panel and select **Edit connections...**
2. On the **Wired** tab, select **System eth0** and click the **Edit...** button
3. Select the **IPv4 Settings** tab
4. On the **Method** drop-down menu, change **Automatic (DHCP)** to **Manual**
5. Under **Addresses** click **Add** and enter the IPv4 address, netmask (in VLSN or CIDR notation), gateway router, and DNS server to use
6. **IMPORTANT:** make sure that **Connect automatically** is checked so the interface starts at boot (rather than when the user logs in), and **Available to all users** is checked so that it is available system-wide
7. Click **Apply** to apply your changes.

It is also possible to configure the network by editing interface configuration files. Interface configuration files control the software interfaces for individual network devices. These files are usually named **/etc/sysconfig/network-scripts/ifcfg-<name>**, where <name> refers to the name of the device that the configuration file controls. The following are standard variables found in the file used for static or dynamic configuration.

Static	DHCP	Any
<b>BOOTPROTO=static</b> <b>IPADDR=192.168.0.250</b> <b>PREFIX=24</b> <b>GATEWAY=192.168.0.254</b> <b>DNS1=192.168.0.254</b>	<b>BOOTPROTO=dhcp</b>	<b>DEVICE=eth0</b> <b>ONBOOT=yes</b> <b>HWADDR=52:54:00:00:00:FA</b> <b>NM_CONTROLLED=yes</b>

Table 5.1. Configuration Options for **ifcfg** file



### Note

If NetworkManager is running, any changes made to the **ifcfg-\*** files take effect immediately.



### Note

If you need to configure static routes, the configuration is stored per interface in **/etc/sysconfig/network-scripts/route-<name>**. Details can be found in the Red Hat Enterprise Linux Deployment Guide, see below.



### Important

NetworkManager runs by default on Red Hat Enterprise Linux 6 and may cause conflicts with your network configuration. If you want to permanently manage the network settings manually, add **NM\_CONTROLLED=no** to the **ifcfg-\*** file for each network interface.

**/etc/sysconfig/network** is used to specify the fully-qualified hostname and may specify a static default route if DHCP is not in use:

```
[root@demo ~]# cat /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=demo.example.com
GATEWAY=192.168.0.254
```

As we saw above, **/etc/resolv.conf** specifies the IP addresses of DNS servers and the search domain.



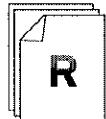
### Important

If DHCP is in use, **/etc/resolv.conf** is automatically rewritten as interfaces are started unless you specify **PEERDNS=no** in the relevant interface configuration files.

Network interfaces can be brought down with the **ifdown eth0** command and brought back up with the **ifup eth0** command, whether managed by NetworkManager or by unmanaged configuration files.

When changing the system configuration you must remember to:

1. Modify a configuration file
2. Restart a service
3. Verify the change



## References

Red Hat Enterprise Linux Deployment Guide

- Section 4.1: Network Configuration Files

Red Hat Enterprise Linux Deployment Guide

- Section 4.2: Interface Configuration Files

Red Hat Enterprise Linux Deployment Guide

- Section 4.4: Configuring Static Routes

Red Hat Enterprise Linux Deployment Guide

- Chapter 5: Network Configuration

**/usr/share/doc/initscripts-\*/sysconfig.txt**



## Practice Quiz

## Viewing and Modifying Network Configuration

- Fill in the below table with the commands, utilities and filenames

Setting Category	View Current Configuration	Change Configuration
IP Address and Subnet Mask		
Routing/Default Gateway		
System Hostname		
Name Resolution		

Table 5.2. Network Configuration from the Command-Line

- What would a dynamic **ifcfg-eth0** contain?
- What would a static **ifcfg-eth0** contain?

## Basic Troubleshooting Process

### Troubleshooting Steps

#### 1. TEST

- Reproduce/verify/characterize the problem
- Monitor the issue
- Gather background information (hardware/software version, etc.)
- Gather diagnostic information (logs, error messages, etc.)
- Determine severity level

#### 2. CHECK

- Look at configuration for evidence of a misconfiguration
- Compare expected settings to intended settings
- Verify proper operation and attachment of local hardware and external resources

#### 3. FIX

- Modify and activate configuration
- Verify by re-running the TEST phase to ensure the problem has been resolved

#### 4. DOCUMENT

- The final wrap-up step that should also be performed



**Practice Exercise**

## Document Network Settings

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

Before we break our network configuration, let us document the current settings on our serverX system.

1. Log in to serverX as root.
2.
  - a. What is its current IP Address?
  - b. What is its current CIDR subnet mask?
  - c. What is its current default gateway?
  - d. What is its current hostname?
  - e. What are its current DNS servers?

## Network Troubleshooting Toolkit

The below table shows how to TEST, CHECK, and FIX each of the network troubleshooting categories:

Category	TEST	CHECK	FIX
IP Address and Subnet Mask	<code>ping</code> , access a service	<code>ip addr</code>	<code>/etc/sysconfig/network-scripts/ifcfg-*</code>
Routing/Default Gateway	<code>traceroute</code>	<code>ip route</code>	<code>/etc/sysconfig/network-scripts/ifcfg-*</code> or provided via DHCP
Name Resolution	<code>host</code>	<code>/etc/hosts</code> and <code>/etc/resolv.conf</code>	<code>/etc/sysconfig/network-scripts/ifcfg-*</code>

Table 5.3. Network Troubleshooting from the Command-Line

### Useful commands and files

- `ping`

```
[root@demo ~]# ping -c 2 instructor.example.com
PING instructor.example.com (192.168.0.254) 56(84) bytes of data.
64 bytes from instructor.example.com (192.168.0.254): icmp_seq=1 ttl=64 time=0.697 ms
64 bytes from instructor.example.com (192.168.0.254): icmp_seq=2 ttl=64 time=0.538 ms

--- instructor.example.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 0.538/0.617/0.697/0.083 ms
```

- `ip addr show eth0`

```
[root@demo ~]# ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:00:fa brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.250/24 brd 192.168.0.255 scope global eth0
        inet6 fe80::5054:ff:fe00:fa/64 scope link
```

```
[root@demo ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="dhcp"
HWADDR="52:54:00:00:00:FA"
NM_CONTROLLED="yes"
ONBOOT="yes"
```

- **/etc/sysconfig/network-scripts/ifcfg-<name>**

```
[root@demo ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="dhcp"
HWADDR="52:54:00:00:00:FA"
NM_CONTROLLED="yes"
ONBOOT="yes"
```

- **traceroute**

```
[root@demo ~]# traceroute -Tn www.redhat.com
traceroute to www.redhat.com (184.85.80.112), 30 hops max, 60 byte packets
 1  192.168.0.254  0.641 ms  0.606 ms  0.590 ms
 2  172.31.35.1  9.829 ms  9.531 ms  9.237 ms
 3  204.60.4.40  27.954 ms  27.726 ms  27.385 ms
 4  66.159.184.226  27.128 ms  49.156 ms  48.291 ms
 5  151.164.92.147  43.256 ms  42.995 ms  42.155 ms
 6  12.122.81.57  60.897 ms  60.041 ms  54.531 ms
 7  75.149.230.169  54.143 ms  75.149.231.45  46.412 ms  192.205.37.34  40.208 ms
 8  68.86.86.45  67.587 ms  54.599 ms  53.381 ms
 9  68.86.86.234  65.540 ms  62.189 ms  53.777 ms
10  68.86.87.166  57.084 ms  55.752 ms  57.154 ms
11  184.85.80.112  55.707 ms  58.702 ms  57.996 ms
```

- **ip route**

```
[root@demo ~]# ip route
192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.250 metric 1
default via 192.168.0.254 dev eth0 proto static
```

- **host**

```
[root@demo ~]# host i
i.example.com is an alias for instructor.example.com.
instructor.example.com has address 192.168.0.254
```

- **dig**

```
[root@demo ~]# dig i.example.com

; <>> DiG 9.7.0-P2-RedHat-9.7.0-5.P2.el6 <>> i.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17644
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;i.example.com.           IN      A

;; ANSWER SECTION:
i.example.com.        86400   IN      CNAME   instructor.example.com.
instructor.example.com. 86400   IN      A       192.168.0.254
```

```
;; AUTHORITY SECTION:  
example.com. 86400 IN NS instructor.example.com.  
  
;; Query time: 2 msec  
;; SERVER: 192.168.0.254#53(192.168.0.254)  
;; WHEN: Mon Dec 13 15:50:21 2010  
;; MSG SIZE rcvd: 86
```

- **/etc/hosts**

```
[root@demo ~]# cat /etc/hosts  
192.168.0.250 demo.example.com demo # Added by NetworkManager  
127.0.0.1 localhost.localdomain localhost  
::1 demo.example.com demo localhost6.localdomain6 localhost6
```

- **/etc/resolv.conf**

```
[root@demo ~]# cat /etc/resolv.conf  
# Generated by NetworkManager  
domain example.com  
search example.com  
nameserver 192.168.0.254
```

### References

Red Hat Enterprise Linux Deployment Guide

- Chapter 4: Network Interfaces

Red Hat Enterprise Linux Deployment Guide

- Chapter 5: Network Configuration

Test

## Criterion Test

Exercise

### Troubleshooting Network Configuration from the Command-line

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

All of the following should be performed on your virtual server, serverX. You will start by running a script that will "break" your network configuration. You will have ten minutes to resolve each of the three problems. Be sure to document what you have found, as we will review at the end.

1. Run the first script to misconfigure your networking:

**lab-break-net 1**

2. Symptom: A web browser is unable to access the web page at `http://instructor.remote.test`
3. Apply the three steps: TEST, CHECK, FIX to identify and resolve the problem.
4. Document what you have found

Use this space for notes



5. Run the second script to misconfigure your networking:

**lab-break-net 2**

6. Symptom: A web browser is unable to access the web page at `http://instructor.remote.test`
7. Apply the three steps: TEST, CHECK, FIX to identify and resolve the problem.
8. Document what you have found

Use this space for notes

9. Run the third script to misconfigure your networking:

**lab-break-net 3**

10. Symptom: A web browser is unable to access the web page at `http://instructor.remote.test`
11. Apply the three steps: TEST, CHECK, FIX to identify and resolve the problem.
12. Document what you have found

Use this space for notes



## Personal Notes



## Unit Summary

### Understanding Network Configuration Files

In this section you learned how to:

- Change network configuration with command-line tools
- Make network configuration changes persistent by editing files

### Basic Troubleshooting Process

In this section you learned how to:

- Employ a systematic approach to troubleshooting system problems

### Network Troubleshooting Toolkit

In this section you learned how to:

- Diagnose and correct network problems so that network access is restored



## **UNIT SIX**

# **MANAGING SIMPLE PARTITIONS AND FILE SYSTEMS**

## **Introduction**

Topics covered in this unit:

- Adding file system space
- Encrypting partitions
- Adding swap space

# Simple Partitions and File Systems

Storage is a basic need of every computer system. Red Hat Enterprise Linux includes powerful tools for managing many types of storage devices in a wide range of scenarios.

**fdisk** is a utility to manage disk partitions. You can view disks and their partitioning by running the utility with the **-l** option and the name of the disk (**fdisk -l /dev/vda**). Changes can be made by running the utility interactively and choosing appropriate menu options (**fdisk -cu /dev/vda**). **-c** disables legacy DOS-compatibility mode and **-u** displays output in sectors (not cylinders, which are obsolete).



## Important

Red Hat Enterprise Linux 6 automatically aligns the first partition to start at sector 2048 instead of sector 63 (the "traditional" start of cylinder 1). This is to ensure maximum performance on new 4 KiB sector hard drives as well as legacy 512 byte sector hard drives, and is compatible with the behavior of other recent operating systems that use the MBR partitioning scheme. Partition misalignment can lead to significant performance loss, so be careful adjusting these settings.

For your virtual server, **serverX**, verify the current storage configuration. Look for information in the output of the following command: **fdisk -l /dev/vda**

Primary Disk:

1. Name: **/dev/vda**
2. Size: **6442 MB**
3. Total sectors: **12582912**
4. Last used sector: **9914367**

```
[root@serverX ~]# fdisk -l
Disk /dev/vda①: 6442 MB②, 6442450944 bytes
16 heads, 63 sectors/track, 12483 cylinders, total 12582912 sectors③
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000a9b12

      Device Boot   Start     End   Blocks  Id  System
/dev/vda1 *       2048   526335   262144   83  Linux
/dev/vda2        526336  9914367④  4694016   8e  Linux LVM
```

- ① Name of disk
- ② Total size of disk
- ③ Total sectors

- ④ Last used sector

## Create a New Partition

```
[root@serverX ~]# fdisk -cu /dev/vda
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 3
First sector (9914368-12582911, default 9914368): Enter
Using default value 9914368
Last sector, +sectors or +size{K,M,G} (9914368-12582911, default 12582911): +1G

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table
WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
[root@serverX ~]# reboot
```

## File System Comparison

- **ext4** is the standard file system for Red Hat Enterprise Linux 6. It is very robust and reliable, and has many features to improve performance for modern workloads.
- **ext2** is an older file system commonly used in Linux; it is simple and reliable, and works well for small storage devices, but is not as efficient as **ext4**.
- **vfat** support covers a family of related file systems (VFAT/FAT16, FAT32) developed for older versions of Microsoft Windows and supported on a wide variety of systems and devices.

## Creating and Using a New File System

1. **mkfs -t filesystem /dev/partition** creates the type of file system requested.
2. **blkid** displays information about the contents of block devices (partitions and logical volumes) including the UUID of the file system.
3. **mkdir /mountpoint** creates a directory to link the new file system to.
4. Add an entry to **/etc/fstab** using the obtained UUID from the **blkid** command:

```
UUID=uuid /mountpoint ext4 defaults 1 2
```

5. Mount the new file system with **mount /mountpoint**.



## Warning

When adding new file systems to **/etc/fstab**, you should use **blkid** to determine its' UUID and mount by UUID. You should *not* mount file systems on simple partitions by standard device name (such as **/dev/sda3**). Disk device names may change depending on the devices visible at boot time, which may cause your system to attempt to mount the wrong file system for the wrong purpose, which at worst could lead to data loss. This is especially important when SAN devices (iSCSI, Fiber Channel) are involved which may be detected by the system in a different order from boot to boot depending on SAN traffic, but it can also matter when removable media such as USB devices may be in use.

Note that Red Hat Enterprise Linux 6 uses UUID instead of LABEL in **/etc/fstab** to reduce the likelihood of naming collisions. The installer no longer uses **e2label** to set labels on Red Hat Enterprise Linux 6 file systems by default.

## Example of File System Creation

```
[root@serverX ~]# mkfs -t ext4 /dev/vda3
[root@serverX ~]# blkid /dev/vda3
/dev/vda3: UUID="a11fad0-2f5b-49e8-ba43-13de7990d3b9" TYPE="ext4"
[root@serverX ~]# mkdir /test
```

Add an entry to **/etc/fstab**:

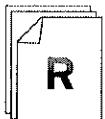
```
UUID="a11fad0-2f5b-49e8-ba43-13de7990d3b9" /test ext4 defaults 1 2
```

Test the mount:

```
[root@serverX ~]# mount /test
```

## Remove an Existing File System

1. Unmount the file system by using **umount /mountpoint**.
2. Remove the corresponding entry in **/etc/fstab**.
3. Remove the mount point directory: **rmdir /mountpoint**.



## References

**fdisk(8), fstab(5), mkfs(8), blkid(8), mount(8)** man pages

Knowledgebase: "How can I create a disk partition on a disk that is greater than 2 TB in size?"  
<https://access.redhat.com/kb/docs/DOC-4282>



Practice Quiz

## Add a New File System

Fill in the blank with the command that accomplishes the task listed.

1. Identify a disk that has some free space

---

2. Create a new partition on that disk

---

3. Update the kernel partition table

---

4. Create a file system on the partition

---

5. Determine the UUID of the file system

---

6. Create a mount point

---

7. Add an entry to the file system table file

---

8. Mount the file system

---

## Enabling Data Privacy with Partition Encryption

LUKS ("Linux Unified Key Setup") is a standard format for device encryption. LUKS encrypts the partition or volume; the volume must be decrypted before the file system in it can be mounted.

### Create a New Encrypted Volume

1. Create a new partition with **fdisk**
2. **cryptsetup luksFormat /dev/vdaN** encrypts the new partition and sets the decryption password
3. **cryptsetup luksOpen /dev/vdaNname** unlocks the encrypted volume /dev/vdaN as /dev/mapper/*name* after you enter the correct decryption password
4. Create an **ext4** file system on the decrypted volume: **mkfs -t ext4 /dev/mapper/name**
5. Create the directory mountpoint and mount the file system: **mkdir /secret ; mount /dev/mapper/name /secret**
6. When finished, **umount /dev/mapper/name** and run **cryptsetup luksClose name** to lock the encrypted volume



### Note

When creating an encrypted partition, it is best to write random data to the raw device before using LUKS to initialize the volume. This may help make attacks on the disk encryption more difficult. Copying data from **/dev/urandom** to the device file for the partition will accomplish this, but it can take a long time.

### Persistently Mount Encrypted Partition

1. **/etc/crypttab** contains a list of devices to be unlocked during system startup.

①**name** ②**/dev/vdaN** ③**/path/to/password/file**

**/etc/crypttab** lists one device per line, with the following space separated fields:

- ① Name device mapper will use for the device
- ② The underlying "locked" device
- ③ Password file to use to unlock the device. If this field is left blank (or set to **none**), the user will be prompted for the decryption password during startup

2. Create an entry in **/etc/fstab** like the following:

**/dev/mapper/name /secret ext4 defaults 1 2**



### Warning

The device listed in the first field of `/etc/fstab` must match the name chosen for the local name to map in `/etc/crypttab`. This is a common configuration error.

3. Create the key file that includes the password. Make sure it is owned by root and the mode is 600. Add the key for LUKS using the following command:

```
[root@serverX ~]# cryptsetup luksAddKey /dev/vda5 /path/to/password/file
```

## Encrypted File System Creation Example

Create a new partition as previously done. We will assume the device is `/dev/vda5`.

```
[root@serverX ~]# cryptsetup luksFormat /dev/vda5
WARNING!
=====
This will overwrite data on /dev/vda5 irreversibly.

Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase: testing123
Verify passphrase: testing123
[root@serverX ~]# cryptsetup luksOpen /dev/vda5 encdisk
Enter passphrase for /dev/vda5: testing123
[root@serverX ~]# mkfs -t ext4 /dev/mapper/encdisk
[root@serverX ~]# mkdir /encdisk
[root@serverX ~]# mount /dev/mapper/encdisk /encdisk
```

To make the disk persistent, start by appending the following to `/etc/fstab`:

```
/dev/mapper/encdisk /encdisk ext4 defaults 1 2
```

Create `/etc/crypttab` and add the following line. This will ask the password every time the machine boots:

```
encdisk /dev/vda5
```

## Automatic Entry of Encryption Password

If you want an automated boot, you must place the password in a text file (which has obvious security implications).

`/etc/crypttab`:

```
encdisk /dev/vda5 /root/encdisk

[root@serverX ~]# echo -n "testing123" > /root/encdisk
[root@serverX ~]# chown root /root/encdisk
[root@serverX ~]# chmod 600 /root/encdisk
```

```
[root@serverX ~]# cryptsetup luksAddKey /dev/vda5 /root/encdisk  
Enter any passphrase: testing123
```



## References

Red Hat Enterprise Linux Security Guide

- Section 3.8: LUKS Disk Encryption

**cryptsetup(8)** and **crypttab(5)** man pages



#### Practice Resequencing Exercise

## Create Encrypted File System

For each of the file or directory names below, write down the number of its definition from the list at the bottom.

- Create a new partition
  - Create an **ext4** file system
  - Format the new partition for encryption
  - Mount the file system on the unlocked device
  - Create an entry in **/etc/fstab**
  - Create a directory to use as a mount point
  - Unlock the encrypted partition
  - Create an entry in **/etc/crypttab**
  - Make LUKS aware of the password file
- 
1. **fdisk**
  2. **cryptsetup luksFormat /dev/vdaN**
  3. **cryptsetup luksOpen /dev/vdaNsecret**
  4. **mkfs -t ext4 /dev/mapper/secret**
  5. **mkdir /secret**
  6. **mount /dev/mapper/secret /secret**
  7. **secret /dev/vdaN/password/file**
  8. **/dev/mapper/secret /secret ext4 defaults 1 2**
  9. **cryptsetup luksAddKey /dev/vdaN/password/file**

## Managing Swap Space

*Swap space* or a swap area is space on the disk drive used as overflow for parts of memory that are not currently being used. This allows the system to make room in main memory for data currently being processed, and provides emergency overflow if the system is at risk of running out of space in main memory.

### Creating and Using an Additional Swap Partition

1. Create a new partition using **fdisk**. Additionally, change the partition type to "**0x82 Linux Swap**" before saving changes with **fdisk**.
2. **mkswap /dev/vdaN** will prepare the partition for use as a swap area.
3. **blkid /dev/vdaN** will determine the UUID.
4. Add the new swap space to **/etc/fstab**:

```
UUID=uuid swap swap defaults 0 0
```

5. **swapon -a** will activate the new swap area.

**swapon -s** will show status of current swap areas.

**swapoff /dev/vdaN** will de-activate that particular swap area.

### Example of Swap Space Creation

Create a new partition and change the type to 82:

```
[root@serverX ~]# fdisk /dev/vda
Command (m for help): n
First sector (12539904-12582911, default 12539904): Enter
using default value 12539904
Last sector, +sectors or +size{K,M,G} (12539904-12582911, default 12582911): Enter
using default value 12582911

Command (m for help): t
Partition number (1-6): 6
Hex code (type L to list codes): 82
changed system type of partition 6 to 82 (Linux swap / Solaris)

Command (m for help): w

[root@serverX ~]# reboot
```

Write the swap signature to the device and find the UUID:

```
[root@serverX ~]# mkswap /dev/vda6
[root@serverX ~]# blkid /dev/vda6
/dev/vda6: UUID="4903c440-ffcb-4404-bc09-505c79c7a412" TYPE="swap"
```

Add an entry to **/etc/fstab**:

```
UUID="4903c440-ffcb-4404-bc09-505c79c7a412" swap swap defaults 0 0
```

Activate the swap space, verify it is available and then deactivate the swap space:

```
[root@serverX ~]# swapon -a
swapon -s
/dev/dm-0          partition      557048  0      -1
/dev/vda6          partition     21496   0      -2
[root@serverX ~]# swapoff /dev/vda6
```

Sizing total swap space should really be based on the memory workload on the system, not the total amount of physical memory present. However, the table below provides some rough rules of thumb for sizing swap space. For more detailed guidance on sizing swap space, see the Knowledgebase article in the references.

<i>System RAM</i>	<i>Recommended Minimum Swap Space</i>
up to 4 GB	at least 2 GB
4 GB to 16 GB	at least 4 GB
16 GB to 64 GB	at least 8 GB
64 GB to 256 GB	at least 16 GB

Table 6.1. Basic Guidance on Swap Space Sizing

## References

Red Hat Enterprise Linux Storage Administration Guide

- Chapter 14: Swap Space

Knowledgebase: "If I add several hundred GB of RAM to a system, do I really need several hundred GB of swap space ?"

<https://access.redhat.com/kb/docs/DOC-15252>

**mkswap(8)** and **swapon(8)** man pages



Practice Exercise

## Create and Use a New Swap Partition

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

Perform the following steps on serverX unless directed otherwise.

1. Start **fdisk** and create a new 256 MB partition. Change the partition type to **swap**.



### Important

To have room for creating additional partitions in the future, if needed, be sure to create an Extended partition beforehand

2. Prepare the new partition for use as swap
3. Determine the UUID
4. Add the new partition to **/etc/fstab**
5. Determine current amount of swap
6. Activate the new swap
7. Verify newly activated swap



Test

## Criterion Test

Case Study

### Managing Simple Partitions and File Systems

***Before you begin...***

Run **lab-setup-storage** on desktopX to prepare serverX for the exercise.

Perform the following steps on serverX unless directed otherwise.

Your department wants to use some unallocated storage on the servers. Create additions to your system according to the following list:

- Create a new partition and **ext4** file system that is 400 MB in size. The file system should persistently mount under **/data**.
- Persistently add a swap partition that is 200 MB in size.
- Create an encrypted device with an **ext4** file system that is 256 MB in size and uses the password **testing123**. The system should prompt for the password at boot and mount the file system to **/test**.

When you are ready to check your work, run **lab-grade-storage** on serverX.



#### Important

To have room for creating additional partitions in the future, if needed, be sure to create an Extended partition beforehand.

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*



## Personal Notes



## Unit Summary

### Simple Partitions and File Systems

In this section you learned how to:

- Create and format a simple partition for data storage

### Enabling Data Privacy with Partition Encryption

In this section you learned how to:

- Enable data privacy with an encrypted partition from the command-line

### Managing Swap Space

In this section you learned how to:

- Create and format a simple partition for swap





## **UNIT SEVEN**

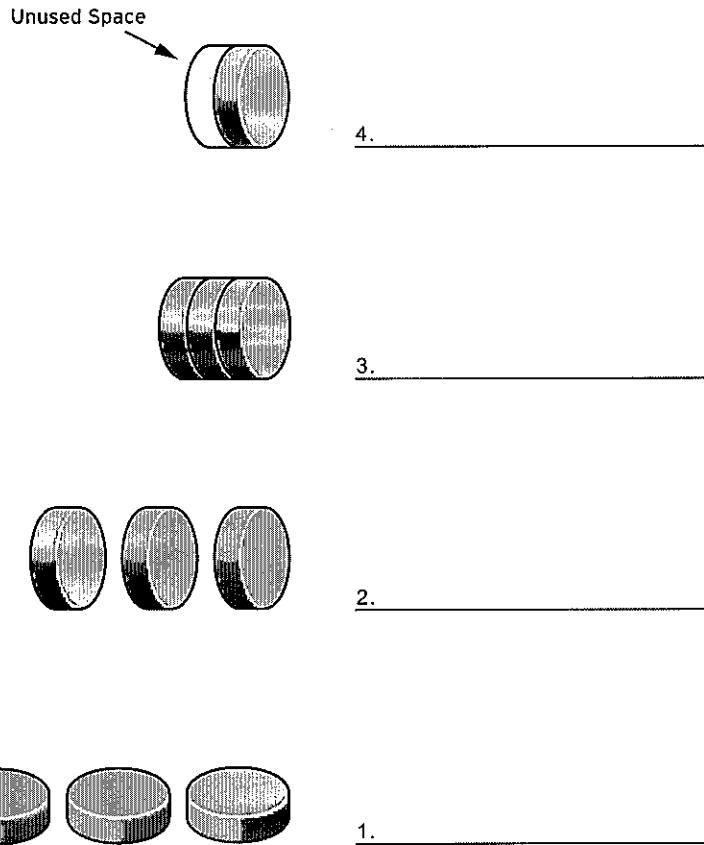
# **MANAGING FLEXIBLE STORAGE WITH LOGICAL VOLUME MANAGER**

## **Introduction**

Topics covered in this unit:

- Review LVM Components
- Implement LVM Storage
- Grow a File System
- Add a Disk
- Snapshot as Backup

## Recognize the Components of LVM



### Review LVM Definitions

- *Physical Partitions or Disks* are the first building block of LVM. These could be partitions, whole disks, RAID sets or SAN disks.
- *Physical Volumes* are the underlying "physical" storage used with LVM. This is typically a block device such as a partition or whole disk. A device must be initialized as an LVM Physical Volume in order to be used with LVM.
- *Volume Groups* are storage pools made up of one or more Physical Volumes.
- *Physical Extents* are small chunks of data stored on Physical Volumes that act as the back end of LVM storage.
- *Logical Extents* map to Physical Extents to make up the front end of LVM storage. By default, each Logical Extent will map to one Physical Extent. Enabling some options will change this mapping. Mirroring, for example, causes each Logical Extent to map to two Physical Extents.

- *Logical Volumes* are groups of Logical Extents. A Logical Volume may be used in the same manner as a hard drive partition.

## Why Use Logical Volumes?

Logical volumes, and logical volume management make it easier to manage disk space. If a file system needs more space, it can be allocated to its logical volume from the free space in its volume group and the file system can be resized. If a disk starts to fail, a replacement disk can be registered as a physical volume with the volume group and the logical volume's extents can be migrated to the new disk.



### References

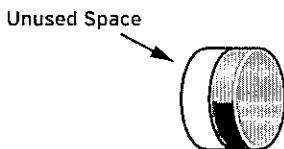
Red Hat Enterprise Linux Logical Volume Manager Administration



## Practice Quiz

### LVM Components

1. Fill in the following graphic with the names of the components.



4. \_\_\_\_\_



3. \_\_\_\_\_



2. \_\_\_\_\_



1. \_\_\_\_\_

2. What are the smallest pieces (chunks or blocks) of the physical volume?

\_\_\_\_\_

3. What is the smallest size you could make a logical volume?

\_\_\_\_\_

4. What references the physical extents of a logical volume?

---

# Implement LVM Storage with Command-line Tools

## Prepare a Physical Volume

1. **fdisk** is used to create a new partition for use with LVM. Always set the Type to **0x8e Linux LVM** on a partition to be used with LVM.



### Note

Alternatively, you can use a whole disk, a RAID array, or a SAN disk.

2. **pvcreate /dev/vdaN** is used to initialize the partition (or other physical device) for use with LVM as a Physical Volume. A header to store LVM configuration data is created directly in the Physical Volume.

## Creating a Volume Group

1. **vgcreate vgname /dev/vdaN** will create a volume group named *vgname* made up of the physical volume */dev/vdaN*. You can specify additional space-delimited physical volumes at the time of creation or add new physical volumes later with **vgextend**.

## Create and Use a New Logical Volume

1. **lvcreate -n lvname -L 2G vgname** creates a new 2 GB logical volume named *lvname* from the available physical extents on *vgname*.



### Important

Different tools will display the logical volume name using either the traditional name, **/dev/vgname/lvname**, or the kernel device mapper name, **/dev/mapper/vgname-lvname**.

2. **mkfs -t ext4 /dev/vgname/lvname** will create an **ext4** file system on the new logical volume.
3. **mkdir /data** makes directory needed as a mount point.
4. Add an entry to the **/etc/fstab** file:

```
/dev/mapper/vgname-lvname /data ext4 defaults 1 2
```

5. Run **mount -a** to mount all the file systems in **/etc/fstab**, including the entry you just added.

## Review LVM Status Information

- `pvdisplay /dev/vdaN`** will display information about the specific physical volume.
- `vgdisplay vgname`** will display information about the specific volume group.

Example **`vgdisplay`** output:

Volume group ---	
VG Name	① <b>vg1</b>
System ID	
Format	lvm2
Metadata Areas	10
Metadata Sequence No	18
VG Access	read/write
VG Status	resizable
MAX LV	0
Cur LV	6
Open LV	6
Max PV	0
Cur PV	5
Act PV	5
VG Size	② <b>7.28 TB</b>
PE Size	③ <b>4.00 MB</b>
Total PE	④ <b>1907727</b>
Alloc PE / Size	⑤ <b>1720587 / 6.56 TB</b>
Free PE / Size	⑥ <b>187140 / 731.02 GB</b>
VG UUID	<b>7FmScA-HJWa-&lt;snip&gt;</b>

- ① The name of the Volume Group
  - ② The total size of the "physical" storage in the Volume Group
  - ③ Physical Extent size
  - ④ Total Physical Extents in Volume Group
  - ⑤ Total Physical Extents used by Logical Volumes
  - ⑥ Physical Extents available
- `lvdisplay /dev/vgname/lvname`** will display information about the specific logical volume.



## References

Red Hat Enterprise Linux Logical Volume Manager Administration

**`lvm(8)`** man page



### Practice Exercise

## Implement LVM and Create a Logical Volume

#### *Before you begin...*

Make sure to run the **lab-setup-lvm** from your desktopX system, which will prepare your serverX system for the practice exercise.

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

All of these steps will be performed on serverX.

1. Create a new partition of 512 MB and prepare it for use with LVM as a Physical Volume.



### Important

To have room for creating additional partitions in the future, if needed, be sure to create an Extended partition beforehand.

2. Create a Volume Group named **shazam** using the Physical Volume created in the previous step.
3. Create and format with **ext4**, a new Logical Volume of 256 MB called **/dev/shazam/storage**.
4. Modify your system such that **/dev/shazam/storage** is mounted at boot time as **/storage**.

# Extend a Logical Volume and Ext4 File System

One benefit of logical volumes is the ability to increase their size without experiencing downtime. Free physical extents in a volume group can be added to a logical volume to "extend" its capacity, which can then be used to extend the file system it contains.

## Growing a Logical Volume Basic Steps

1. Verify available space in the

2. Extend the

3. Extend the

### Extending the Logical Volume and File System

1. Verify the current size of the mounted file system `/data`:

```
# df -h /data
```

2. Verify that there are sufficient "Physical Extents available" for use:

```
# vgdisplay vgname
```

Refer to the sample output in the "Review LVM Status Information" section found earlier in this unit to see how to identify available free physical extents.

3. Extend the logical volume using some or all of the available extents:

```
# lvextend -l 128 /dev/vgname/lvname
```

4. Grow the associated file system mounted on `/data`:

```
# resize2fs -p /dev/vgname/lvname
```

The `-p` option displays progress during the operation.



#### Note

- The file system can remain mounted and be used while `resize2fs` is being run.



## Important

A common mistake is to run **lvextend** but forget to run **resize2fs**.

5. Verify the new size of the mounted file system **/data**:

```
# df -h /data
```

## Reducing a File System and Logical Volume

This process is similar to extending, but *in reverse*: **resize2fs**, then **lvreduce**.



## Warning

It is essential you have a solid backup before undertaking a reduction in the logical volume, as typographical errors in the command line can cause data loss.

1. While extending a logical volume can be done while the file system is in use, reducing an **ext4** file system must be done offline.  
**umount /data** to unmount the file system you want shrink.
2. **fsck -f /dev/mapper/vgname-lvname** to verify that all file system data structures are clean prior to resizing.
3. **resize2fs -p /dev/mapper/vgname-lvname 512M** will resize the file system to be 512 MB, presuming that the logical volume is larger than 512 MB.  
Note: If you omit the size from the **resize2fs** command, it defaults to the size of the logical volume, perfect for extending the logical volume like done previously.
4. **lvreduce -L 512M /dev/mapper/vgname-lvname** will shrink the logical volume to 512 MB.



## Warning

**lvreduce** has no knowledge of your file system data structures, and, without warning, will discard elements of your file system if you did not first use **resize2fs** to make the file system *smaller* than the intended logical volume size.

5. **mount -a** will remount all the file systems listed in **/etc/fstab**, including your now smaller logical volume assuming it is listed in



## References

**Red Hat Enterprise Linux Logical Volume Manager Administration**

**lvm(8) man page**

[Red Hat Enterprise Linux Logical Volume Manager Administration] [lvm(8) man page]



Practice Exercise

## Extend a Logical Volume

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

All of these steps will be performed on serverX.

1. Determine the amount of free space in Volume Group **shazam**.
2. Extend the logical volume **/dev/shazam/storage** with *half* the available extents in the volume group using command-line tools.
3. Extend the file system mounted on **/storage** using command-line tools.

# Extending and Reducing a Volume Group

When the logical volumes in a volume group use all of the volume group's free physical extents, they cannot be extended without adding additional space to the volume group. Thankfully additional physical volumes can be created and added to a volume group to "extend" its capacity.

Another benefit of using LVM is that data can be moved between physical storage devices without user downtime. For example, data can be moved from a slower disk drive to a new, faster disk drive. This allows a system administrator to remove the unused physical storage device from a volume group, in this case the slow disk drive.

## Extending a Volume Group

1. As in the creating a new volume group, a new partition must be created and prepared for use as an LVM Physical Volume.

Use **fdisk** to create a new partition and take care to set the Type to **0x8e Linux LVM**.

Use **pvcreate /dev/vdaN** to initialize the partition for use with LVM as a Physical Volume.

2. **vgextend vgname /dev/vdaN** is used to add the new Physical Volume, **/dev/vdaN**, to an existing Volume Group, **vgname**.
3. Use **vgdisplay** to confirm additional "Physical Extents available".

## Reducing a Volume Group

1. **pvmove /dev/vdaN** is used to relocate any physical extents used on **/dev/vdaN** to other Physical Volumes in the Volume Group. This is only possible if there are enough available extents in the Volume Group and if all of those come from other Physical Volumes.



### Warning

Before using **pvmove**, it is recommended to back up data on logical volumes in the volume group. An unexpected power loss during the operation may leave the volume group in an inconsistent state.

2. **vgreduce vgname /dev/vdaN** is used to remove the Physical Volume **/dev/vdaN** from the Volume Group **vgname**.



## References

Red Hat Enterprise Linux Logical Volume Manager Administration

**lvm(8), pvmove(8), and vgreduce(8) man pages**



**Practice Exercise**

## Extend a Volume Group

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

All of these steps will be performed on serverX.

1. Create a new 512 MB partition and prepare it for use with LVM as a Physical Volume.



### Important

To have room for creating additional partitions in the future, if needed, be sure to create an Extended partition beforehand.

2. Extend the Volume Group **shazam** by adding the Physical Volume created in the previous step.

# Create a Snapshot to Facilitate Data Backup

Snapshot logical volumes are another flexible feature of LVM storage. An LVM snapshot is a logical volume that temporarily preserves the original data of a changing logical volume. The snapshot provides a static view of the original volume so its data can be backed up in a consistent state.

## Determining Snapshot Size

1. Expected rate of \_\_\_\_\_
2. Required snapshot \_\_\_\_\_

The snapshot volume need only be large enough to store the data that will change while it exists.

If more data changes than the snapshot can hold, the snapshot will automatically become unusable. (The original volume will remain unharmed, and the dead snapshot will still need to be unmounted and removed from the volume group manually.)

## Creating and Using a Snapshot for Backup

1. Create a new snapshot volume called ***snaplvname*** of **/dev/vgname/lvname** that is **20 MB** in size.

```
# lvcreate -s -n snaplv -L 20M /dev/vgname/lvname
```

2. If your backup software requires it, mount the snapshot and point the backup program to the new mountpoint:

```
# mkdir /snapmount  
# mount -o ro /dev/vgname/snaplv /snapmount
```

3. Verify the status of the snapshot logical volume:

```
# lvs /dev/vgname/snaplv  
LV      VG      Attr  LSize  Origin Snap%  Move Log Copy% Convert  
snaplv vgname swi-a- 224.00m origlv   0.26
```

4. When done with the snapshot, unmount and remove it:

```
# umount /snapmount  
# lvremove /dev/vgname/snaplv
```



## References

Red Hat Enterprise Linux Logical Volume Manager Administration

**lvm(8)** man page



Practice Exercise

## Creating an LVM Snapshot

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

Compare the contents of our existing logical volume, **/dev/shazam/storage**, to a new snapshot volume, **/dev/shazam/storagesnap**, while making changes to the original volume.

All of these steps will be performed on serverX.

1. Copy the file **/usr/share/dict/linux.words** to **/storage** so you have some data to compare.
2. Create a new 20 MB snapshot logical volume of **/dev/shazam/storage** called **storagesnap**.
3. Manually mount **/dev/shazam/storagesnap** read only at **/storagesnap**.
4. List the contents of **/storagesnap** and note that they are the same as **/storage**.
5. Delete the file **/storage/linux.words** and note that it still exists in **/storagesnap**.
6. Clean up: unmount **/storagesnap**, remove the directory, and delete the **storagesnap** logical volume.

Test

## Criterion Test

### Case Study

## LVM Case Study

#### *Before you begin...*

Run **lab-setup-lvm** on desktopX to prepare serverX for the exercise.

Allison needs to store data for her business. Her customer database is currently 256 MB in size. The data in the database changes about 10 MB per hour on a typical day. The backup software takes 10 minutes to complete a full run.

Create a new Volume Group called **allison** with enough space for both a 512 MB volume and a snapshot of that volume for the backup software.

Once the volume group is created, create within it a 512 MB logical volume for Allison's customer database called **custdb**. Also create a snapshot volume of Allison's customer database called **custdbsnap** for her backup software.

When you are ready to check your work, run **lab-grade-lvm** on serverX.

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*



## Personal Notes



## Unit Summary

### Recognize the Components of LVM

In this section you learned how to:

- Identify the basic building blocks of Logical Volume Manager

### Implement LVM Storage with Command-line Tools

In this section you learned how to:

- Create new physical volumes, volume groups and logical volumes via command-line tools
- Review LVM status information

### Extend a Logical Volume and Ext4 File System

In this section you learned how to:

- Extend a logical volume and corresponding file system to satisfy growing data needs

### Extending and Reducing a Volume Group

In this section you learned how to:

- Add new physical volumes to an existing volume group
- Remove an existing physical volume from a volume group

### Create a Snapshot to Facilitate Data Backup

In this section you learned how to:

- Use temporary LVM snapshots to facilitate data backups, minimizing service downtime



## **UNIT EIGHT**

# **ACCESSING NETWORK FILE SHARING SERVICES**

## **Introduction**

Topics covered in this unit:

- Mount Network Shares
- Automount Network Shares

## Mount Network File Systems

A *network file system* is a file system that, instead of being provided by a block device like a hard drive, is provided by a network attached storage server to multiple hosts over a network. Clients access the remote storage through a special file system protocol and format.

There are two primary protocols which are used in Linux to access network file systems: NFS and CIFS. NFS, the **Network File System**, acts much like a standard file system for Linux, UNIX, and similar operating systems. CIFS, the **Common Internet File System**, is the standard network file system for Microsoft Windows systems.

Whether the shared file system you want to use is based on NFS or CIFS, the same three basic steps apply if you want to mount it and access it on a Linux system.

### Three Basic Steps for Accessing a Network Share

1. *Identify* the remote share to access.
2. *Determine mount point* where it should be mounted and create the mount point's empty directory.
3. *Mount* the network file system with an appropriate command or configuration change.

## NFS: Network File System

NFS, the **Network File System**, is an Internet standard protocol used by Linux, UNIX, and similar operating systems, as their native network file system. It is an open standard under active extension which supports native Linux permissions and file system features.

Red Hat Enterprise Linux 6 supports NFSv4 (version 4 of the protocol) by default, and falls back automatically to NFSv3 and NFSv2 if that is not available. For all versions of NFS to work on the client, two services should be enabled, **rpcbind** and **nfslock**. NFSv4 uses the TCP protocol to communicate with the server, while older versions of NFS may use either TCP or UDP.

There are two ways to *identify* the NFS exports, or file shares, provided by a server. If the server supports NFSv3 or NFSv2, you may use the **showmount -e server** command to get a list of exports. If the server supports NFSv4, you may mount the / export on an empty directory and browse the contents of all exported file systems.

### Commands for Accessing an NFS Export

1. *Identify*:

```
showmount -e nfsserver.domain
```

2. *Determine mount point*:

```
mkdir /remote1
```

3. *Mount*:

```
mount nfsserver.domain:/exported/path /remote1
```



### Note

To identify shares for NFSv4, try mounting the / export and browse, or continue to use **showmount -e** if the server also supports NFSv3 and/or NFSv2.

```
[root@host ~]# mkdir /allremote1  
[root@host ~]# mount nfsserver.domain:/ /allremote1  
[root@host ~]# ls /allremote1
```

Use this space for notes

## CIFS: Common Internet File System

CIFS is the native network file system for Microsoft Windows operating systems. Linux systems can mount and access CIFS file shares as normal network file systems. However, since CIFS is built around the NTFS file system permissions model and its own authentication system, not everything in the CIFS protocol maps well to how the same things are done in Linux.

The **smbclient** utility, included with the *samba-client* RPM package, can be used to identify CIFS shares provided by Windows or Samba file servers. It works much like clicking on **My Network Places** in Microsoft Windows does. Then the **mount** command can be used to mount the share.

### Commands for Accessing a CIFS Share

1. *Identify:*

```
smbclient -L cifsdomain
```

2. *Determine mount point:*

```
mkdir /remote2
```

3. *Mount:*

```
mount //cifsdomain/sharename /remote2
```



### Note

The **mount** command above will authenticate to the CIFS share using the name of the current user. If you want to specify a different Windows username, use the option **-o username=user** on the **mount** command line.



### Workshop

## Accessing a CIFS Share

*Follow along with the instructor as you complete the steps below.*

In this workshop, we will access a CIFS-based share on **instructor.example.com**.

1. *Identify:*

```
[root@serverX ~]# smbclient -L cifsserver.domain
Enter root's password: <Enter>
Anonymous login successful
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.4-68.el6]

[  Sharename      Type      Comment
-----  -----
[  ftp           Disk      Instructor Public FTP
[  IPC$          IPC       IPC Service (Instructor Samba Server)
Anonymous login successful
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.5.4-68.el6]

[  Server        Comment
-----  -----
[  INSTRUCTOR    Instructor Samba Server

[  Workgroup     Master
-----  -----
[  MYGROUP      
```

2. *Determine mount point:*

```
[root@serverX ~]# mkdir /remote2
```

3. *Mount:*

```
[root@serverX ~]# mount -o username=guest2000+X //instructor.example.com/ftp /remote2
Password: password
[root@serverX ~]# ls /remote2
EXAMPLE-CA-CERT example-ca.crt gls lost+found materials rhel6 solutions
```



## References

Red Hat Enterprise Linux Storage Administration Guide

- Section 10.2: NFS Client Configuration

**nfs(5), mount(8), mount.nfs(8) and mount.cifs(8) man pages**



Practice Exercise

## Accessing an NFS File Server

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

These steps should be performed on serverX.

After verifying the availability of an NFS share, create the necessary mountpoint and temporarily mount the share.

1. Verify that **/var/ftp/pub** is an available NFS share on the server **instructor.example.com**.
2. Create the directory **/server** for use as a mount point.
3. Devise and execute a command to mount the NFS share to the mount point.
4. List the contents of the mount point to verify that it is the contents of the NFS share from **instructor.example.com**
5. Cleanup by unmounting the share

# Automatically Mount Network Storage

Using the **mount** command requires **root** privileges to connect to network shares. As an alternative, we could add entries to **/etc/fstab**, but then the connections to the network servers would be active all the time.

The automounter, or **autofs**, service can be configured to mount network shares "on demand", when a program attempts to access a file on the network share. The automounter will then unmount the share once it is no longer in use, after a certain amount of inactivity.



## Note

The interval of inactivity defaults to five minutes, but can be changed globally in **/etc/sysconfig/autofs**.

```
[root@host ~]# grep TIMEOUT /etc/sysconfig/autofs  
# TIMEOUT - set the default mount timeout (default 600).  
TIMEOUT=300  
...output omitted...
```

Normally, we want to allow the network share to stay mounted for a short period of inactivity in case it is used again shortly. This avoids unnecessary mount/unmount cycles.

In this section, we will look at two ways in which the automounter can be used. Firstly, by using the special **/net** automount mount point, and then secondly by manually configuring *indirect automount maps*.

## Special Map **/net**

- By default, with the **autofs** service running, a special directory named **/net** will exist, but will appear to be empty.
- Accessing the non-existent directory **/net/nfsserver.domain** will cause the automounter to create that subdirectory and display all the NFS exports available from that NFS server. This is sometimes called "browsing" the shares:

```
[User@host ~]$ cd /net/nfsserver.domain  
[User@host nfsserver.domain]$ ls  
exports
```

- Once any files and directories underneath **/net/nfsserver.domain** stop being used and the timeout expires, **autofs** unmounts the shares and removes the empty **/net/nfsserver.domain** subdirectory.



## Workshop Using /net

Follow along with the instructor as you complete the steps below.

1. Identify and record the path in /net needed to access the export /var/ftp/pub on server **instructor.example.com**:
- 

2. Validate the path by using the **cd** to access it on your serverX.

```
[student@serverX ~]$ cd /net/instructor.example.com/var/ftp/pub
[student@serverX pub]$ ls
EXAMPLE-CA-CERT example-ca.crt gls lost+found materials rhel6 solutions
```

Use this space for notes

---

## Indirect Maps

Instead of using the **/net** map, the system administrator can also manually configure an arbitrary directory which mounts particular shares on its subdirectories "on demand" when accessed. One way to do this is to use the automounter with *indirect maps*.

The arbitrary directory which contains the mount points managed by the automounter will be configured in the **/etc/auto.master** configuration file. Which subdirectories will be created as mount points in that directory and what will be mounted on them is configured in a second file. The location of this second file is also specified in the **/etc/auto.master** file in the entry for the parent directory.

### Example

- Indirect maps use a two-tier configuration file syntax. The top level file, **/etc/auto.master**, will have one line for each "parent" directory to manage and the name of an individual second configuration file that contains the sub-directory *mount point* and *network share*.

```
[root@host ~]# cat /etc/auto.master
❶ /demo ❷ /etc/auto.demo
[root@host ~]# cat /etc/auto.demo
❸ public ❹ -ro ❺ nfsserver.domain:/exported/path
```

- ❶ Parent directory of the mountpoint. This directory is visible on the system at all times and is being monitored by the **autofs** service to determine the "need" to create/mount the subdirectory mountpoint.

- ❷ Individual configuration file containing a list of subdirectory mountpoints managed under this parent directory by the **autofs** service.
- ❸ Subdirectory mountpoint. This directory is normally invisible until directly named/ accessed, after which the **autofs** service creates it and mounts the share.
- ❹ Mount options to be used when mounting the network share.
- ❺ Network share to be mounted when subdirectory mountpoint is accessed.

```
[user@host ~]$ cd ❶/demo/public  
[user@host public]$
```

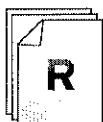
- ❶ Mountpoint that upon access will be created and mounted to the defined network share. Like the above **/net** examples, after a period of inactivity, this mountpoint will be unmounted and deleted.



### Note

Indirect maps also support *metacharacters* for wildcarding mountpoint and share definitions. This is useful when automounting user home directories from a network file server. We will cover this topic in an upcoming unit.

Use this space for notes



### References

**auto.master(5), autofs(5)** man pages for configuration

**autofs(8), automount(8)** man pages for daemon operation



Practice Exercise

## Automount NFS Share with Indirect Maps

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

These steps should be performed on serverX.

You will now set up the automounter to automatically mount an NFS export on a specified directory on demand.

- The NFS server is **instructor.example.com**
- The NFS export on the server is **/var/ftp/pub**
- The automatic mount point on your station should be **/server/public**
- Use an indirect map (not **/net**) to implement this
- Validate that the automount works by changing directory to the mount point and accessing the files in the share

Perform the following steps:

1. Create/modify **autofs** service configuration files.
2. Reload the automounter.
3. Access the share.

Test

## Criterion Test

Case Study

### Automounting NFS Shares

These steps should be performed on serverX.

- Your company has taken on a new client, the Organization of Secret Hidden Undertakings (or OSHU).
- The company has a new NFS server with shares for storing files related to "special" clients: **instructor.example.com**
- The share for this client is: **/var/nfs/oshu**
- Configure your workstation such that autofs automatically mounts that share as: **/special/oshu**
- List the contents of **/special/oshu** to verify that you see the following files:  
**supertopsecret.txt**  
**ultrashh.txt**

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*



## Personal Notes



## Unit Summary

### Mount Network File Systems

In this section you learned how to:

- Manually access available NFS and CIFS shares from the command-line

### Automatically Mount Network Storage

In this section you learned how to:

- Use **autofs** default special map, **/net** to access an available NFS share
- Configure **autofs** using indirect maps for an available NFS share





## **UNIT NINE**

# **MANAGING USER ACCOUNTS**

## **Introduction**

Topics covered in this unit:

- User Definition
- Manage Local Users
- Password Expiration

## What Is a User?

Every process (running program) on the system runs as a particular user. Every file is owned by a particular user. Access to files and directories are restricted by user. The user associated with a running process determines the files and directories accessible to that process.

To view the user associated with a process, include the **u** option to the **ps** command. The first column shows the user name:

```
[root@serverX ~]# ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root        1  0.0  0.2 19232 1392 ?        Ss Jan24  0:00 /sbin/init
root        2  0.0  0.0     0     0 ?        S    Jan24  0:00 [kthreadd]
gdm       1913  0.0  0.3 132372 1972 ?        S    Jan24  0:00 /usr/libexec/gv
student   29261  1.7  0.4 97588 2100 ?        S   20:38  0:00 sshd: student@pts/0
student   29281  0.0  0.2 106008 1240 pts/0      S   20:38  0:00 /bin/sh /usr/bi
```

To view the user associate with a file or directory, use the **ls -l** command. The third column shows the user name:

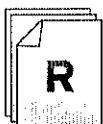
```
[root@serverX ~]# ls -l /tmp
drwx----- . 2 gdm      gdm      4096 Jan 24 13:05 orbit-gdm
drwx----- . 2 student  student  4096 Jan 25 20:40 orbit-student
-rw-r--r-- . 1 root     root     23574 Jan 24 13:05 postconf
```

Linux users are defined in databases. By default, systems use a simple "flat file", the **/etc/passwd** file, to store information about local users. The format of **/etc/passwd** follows (seven colon-separated fields):

**1** **username**:**2** **password**:**3** **UID**:**4** **GID**:**5** **GECOS**:**6** **/home/dir**:**7** **shell**

User Attributes:

- 1** *username* is a mapping of UID to a name for the benefit of human users
- 2** *password* is where, historically, passwords were kept in an encrypted format. Today, they are stored in a separate file called **/etc/shadow**.
- 3** *UID* is a user ID, a number that identifies the user at the most fundamental level
- 4** *GID* is the primary group ID, by default, a number associated with all newly created files, potentially for collaboration
- 5** *GECOS* field stores arbitrary text, normally the real name, office number, or phone number of the user
- 6** */home/dir* is the location of the user's personal data and configuration files
- 7** *shell* is the program that executes as the user logs in, and, if a shell, like **/bin/bash**, provides the user's command line prompt



## References

Red Hat Enterprise Linux Deployment Guide

- Chapter 15: Users and Groups

**passwd(5) man page**

# Managing Local Users

A number of command-line tools can be used to manage local user and group accounts.

## **useradd** Creates Users

- **useradd *username*** sets reasonable defaults for all fields in **/etc/passwd** when run without options
- **useradd** does not set any valid password by default, the user cannot log in.
- **useradd --help** will display the basic options that can be used to override the defaults

## **userdel** Deletes Users

- **userdel *username*** removes the user from **/etc/passwd**, but, by default, leaves the home directory intact.
- **userdel -r *username*** removes the user and the user's home directory



### Warning

When a user is removed with **userdel** without the **-r** option specified the system will have files that are owned by an unassigned user ID number. This can also happen when files created by a deleted user exist outside their home directory. This situation can lead to information leakage and other security issues.

The **useradd** command assigns new users the first free UID number (unless one is explicitly specified with the **-u *UID*** option). This is how the information leakage can occur. If the first free UID number had been previously assigned to a user account which has since been removed from the system, the old user's UID number will get reassigned to the new user giving the new user ownership of the old user's remaining files! The following demonstrates this situation:

```
[root@serverX ~]# useradd prince
[root@serverX ~]# ls -l /home
drwx----- 4 prince  prince  1024 Dec 23 12:30 prince
[root@serverX ~]# userdel prince
[root@serverX ~]# ls -l /home
drwx----- 4      500    500   1024 Dec 23 12:30 prince
[root@serverX ~]# useradd bob
[root@serverX ~]# ls -l /home
drwx----- 4 bob    bob    1024 Dec 23 12:31 bob
drwx----- 4 bob    bob    1024 Dec 23 12:30 prince
```

Notice that **bob** now owns all files that **prince** once owned. The best solution to this problem is to remove all "unowned" files from the system when the user that created them is deleted.

## **id** Displays User Information

- **id** will display user information, including the user's UID number and group memberships.

- **id *username*** will display user information for *username*, including the user's UID number and group memberships.

## passwd Sets Passwords

- **passwd *username*** can be used to either set the user's initial password or change that user's password.

Use this space for notes

---

## UID Ranges

- **UID 0** is **root** and has special privileges
- **UID 1-499** are "system users" by convention - generally non-interactive service accounts
- **UID 500+** are "regular users" for people to use for interactive access to the machine



## References

- Red Hat Enterprise Linux Deployment Guide
  - Section 15.2: User and Group Management Tools

**useradd(8), usermod(8), userdel(8), groupadd(8), groupdel(8) man pages**



Practice Exercise

## Create Users Using Command-line Tools

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

Perform the following steps on serverX unless directed otherwise.

Create a number of users on your serverX system, setting an initial password (recording in the blanks below).

1. Log into serverX as *root*.
2. Add the user *juliet*.
3. Confirm that *juliet* has been added using the **id** command.
4. Confirm that *juliet* has been added by examining the **/etc/passwd** file.
5. Use the **passwd** command to initialize *juliet*'s password and write down the password here:  

---

6. Continue adding the remaining users from the list below Remember to set an initial password and write it down next to each username:

- faraday \_\_\_\_\_
- jack \_\_\_\_\_
- kate \_\_\_\_\_
- james \_\_\_\_\_
- walt \_\_\_\_\_
- ben \_\_\_\_\_
- claire \_\_\_\_\_
- hugo \_\_\_\_\_
- elvis \_\_\_\_\_

# Managing Passwords

Historically, passwords were stored in the **/etc/passwd** file. However, **/etc/passwd** must be world readable because commands such as **ls** need to access that file to map UIDs to user names.

Passwords were migrated to a more secure **/etc/shadow** file where several different password encryption algorithms are supported. As long as encrypted passwords are being stored in a dedicated file, password aging policy and data can be stored as well.

What 3 pieces of information are stored in a password hash?

**\$1\$gCjLa2/Z\$6Pu0EK0AzfCjxjv2hoL0B/**

1. **1** - The hashing algorithm (1 indicates MD5 hash)
2. **gCjLa2/Z** - The salt used to encrypt the hash
3. **6Pu0EK0AzfCjxjv2hoL0B/** - The encrypted hash



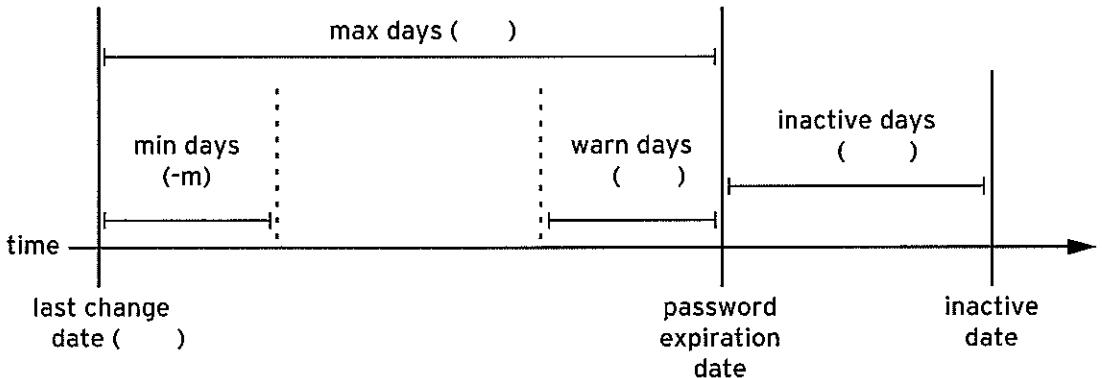
## Note

Red Hat Enterprise Linux 6 supports two new strong password hashing algorithms, SHA-256 (algorithm 5), and SHA-512 (algorithm 6). These may be enabled as the default for **/etc/shadow** using **system-config-authentication** to select it from the **Password Hashing Algorithm** drop-down menu on the **Advanced Options** tab.

## /etc/shadow Fields

1. Username
2. Password hash
3. Date of last password change (number of days since 1970.01.01)
4. Minimum password age (in days, 0 = no minimum age requirement)
5. Maximum password age (in days)
6. Password warning period (in days, 0 = no warning given)
7. Password inactive period (in days)
8. Account expiration (number of days since 1970.01.01)

The following diagram relates the relevant password aging parameters which can be adjusted using **chage** to implement a password aging policy.



As your instructor discusses these parameters, fill in the parentheses in the diagram above with the relevant (short) **chage** command line switch.

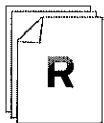
As an example, **-m** has been added to the *min days* parameter to get you started.

```
# chage -m 0 -M 90 -W 7 -I 14 username
```

**chage -d 0 username** will force a password update on next login.

**chage -l username** will list a username's current settings.

**usermod** can modify an account, including "locking" with the **-L** option.



## References

Red Hat Enterprise Linux Deployment Guide

- Section 15.6: Shadow Passwords

**chage(1)**, **usermod(8)**, **shadow(5)**, **crypt(3)** man pages



Practice Quiz

## Account Maintenance

1. What command would lock **elvis**'s account?
2. What command would then unlock it?
3. What command would cause **elvis**'s account to expire on March 15th, 2012?



Test

## Criterion Test

Exercise

### Managing Password Aging Policies

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

Your instructor will divide you into small groups. Within each group, discuss which password aging policies would be appropriate for *professors* (who will be using the machine for a long time), *graduate students* (who will be using the machine for a few years), and *summer interns* (who will only be using the machine for the summer).

- Professors: faraday, juliet
- Graduate Students: jack, kate, james
- Summer Interns: walt, ben, claire, hugo

Assign a password policy to each group of users on serverX.

1. For each group of users (professors, grads, and interns), determine a password aging policy which would be appropriate, including:
  - Account expiration dates (if appropriate).
  - Time before passwords must be changed.
  - Time before unchanged passwords force an account to go inactive.
2. Once determined, use **chage** to implement your policy for the users added in the previous section, according to their role.
3. Additionally, force all users to change their password on first login.



## Personal Notes



## Unit Summary

### What Is a User?

In this section you learned how to:

- Understand what a user account is on a Linux system
- Identify fields of the **/etc/passwd** file

### Managing Local Users

In this section you learned how to:

- Add and remove user accounts from the command-line
- Set and/or change user passwords

### Managing Passwords

In this section you learned how to:

- Customize password aging policies for the users to meet organizational security requirements



## **UNIT TEN**

# **NETWORK USER ACCOUNTS WITH LDAP**

## **Introduction**

Topics covered in this unit:

- LDAP client configuration
- Automounter metacharacters

## Network Authentication Using an LDAP Server

So far in this class, we have looked at local user accounts managed through local files (such as `/etc/passwd`) on each machine. However, it is difficult to coordinate local user accounts on many systems.

In this section, we will look at how to set up a machine as a client, to use network user accounts that are provided by an existing LDAP directory service. This allows the LDAP directory to be our central authority for all network users and groups in our organization.

*User account information* determines the characteristics and configuration of the account. *Authentication methods* are used to determine if someone trying to log in should get access to the account. *Network directory services* can provide both user account information and authentication methods.

*LDAP directory servers* can be used as a distributed, centralized, network user management service. Directory entries are arranged in a tree structure that can be searched. The *base DN (Distinguished Name)* is the base of the tree that will be searched for directory entries for users and groups.

### Key Elements for LDAP Client Configuration

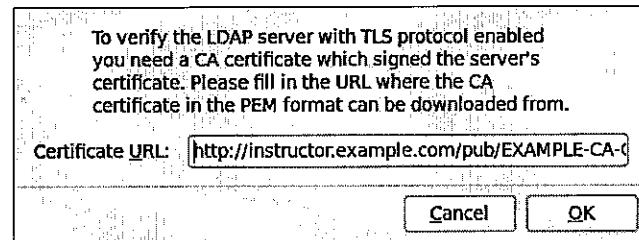
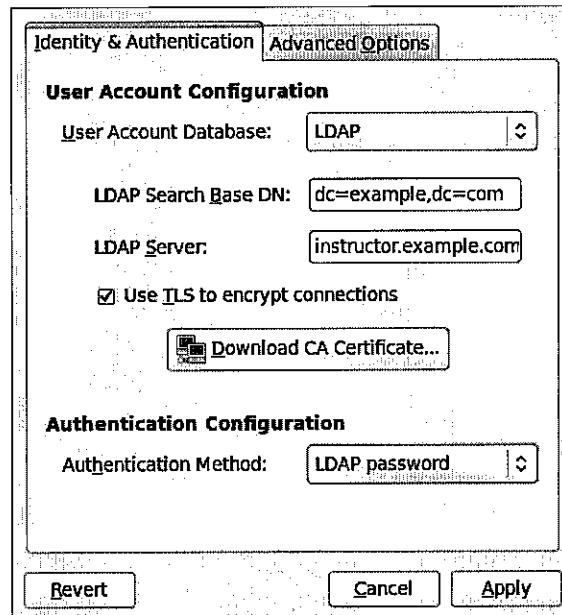
1. Server's fully-qualified hostname
2. Base DN to search for user definitions
3. The certificate authority ("CA") certificate used to sign the LDAP server's SSL certificate

Use this space for notes

---

You should ensure that the **directory-client** yum package group is installed, which includes the packages `sssd`, `authconfig-gtk`, and `oddjob-mkhomedir`, before you begin.

**System → Administration → Authentication** or **system-config-authentication** can be used to modify the configuration of *Identity & Authentication*.



**system-config-authentication** will automatically turn on the **sssd** service which will look up and cache LDAP user information and authentication credentials for the client. If the LDAP server is unavailable but **sssd** is working, the system may be able to authenticate and get information about network users from the **sssd** cache.

Use **getent passwd username** to verify the account information being used. This works whether the user is a local user defined in **/etc/passwd** or a network user from an LDAP service. The command will always show the definition that is actually being used by the system if there is any duplication between local users and network users. By default, the local user definition overrides the network user definition.



### Note

In Red Hat Enterprise Linux 6, **getent passwd** (without specifying a username) will only dump out local usernames by default, it will not dump out the list of all LDAP users as it did in Red Hat Enterprise Linux 5. This is done for performance reasons; see **sssd.conf(5)** under the **enumerate** option for details. (This behavior can be changed by setting **enumerate = True** in the **[domain/default]** section of **/etc/sssd/sssd.conf**.)



## Important

When using LDAP password as your authentication method, you *must* select and configure **Use TLS to encrypt connections**. This is to prevent clear-text passwords from being sent to the LDAP server over the network for authentication.

This is a change from Red Hat Enterprise Linux 5, which would allow the insecure use of LDAP password authentication without TLS. In RHEL 6, you may still use LDAP without TLS if you are using LDAP to get user information only. (For example, you may be using Kerberos for password authentication.) It is better practice to always use TLS.



## References

Red Hat Enterprise Linux Deployment Guide

- Chapter 8: Authentication Configuration

**`system-config-authentication(8)`, `sssd(8)`, and `sssd.conf(5)` man pages**



Practice Quiz

## LDAP Client Configuration

1. What seven pieces of information are typically provided by *User account information* services?
  
2. What "other" type of information can be provided by a *network directory service*?
  
3. What are the three pieces of information a client machine needs to be configured to get user information from an LDAP directory service?
  
4. What does the command **getent passwd ldapuser1** do? Why is this useful?

## Network Mounting Home Directories

Recall that mounting network shares requires four pieces of information: hostname, sharename, mountpoint and mount options.

1. Use **showmount -e nfsserver.domain** to get the exported path that when combined with the hostname gives us the *sharename*.

```
[root@serverX ~]# showmount -e instructor.example.com
Export list for instructor.example.com:
/home/guests 192.168.0.0/255.255.255.0
/var/nfs     192.168.0.0/255.255.255.0
/kickstart   192.168.0.0/255.255.255.0
/var/ftp/pub 192.168.0.0/255.255.255.0
```

2. Use **getent passwd username** to get the needed home directory *mountpoint*.

```
[root@serverX ~]# getent passwd ldapuser1
ldapuser1:*:1701:1701:LDAP Test User 1:/home/guests/ldapuser1:/bin/bash
```



### Note

**getent passwd** only shows the local accounts by default. You can display a single account if you explicitly add the name (as above). If you want to show all of the available accounts (including LDAP accounts), add the following to **/etc/sssd/sssd.conf** beneath the **[domain/default]** section:

```
enumerate = True
```

Restart the **sssd** service:

```
[root@serverX ~]# service sssd restart
[root@serverX ~]# getent passwd
...
ldapuser10:*:1710:1710:LDAP Test User 10:/home/guests/ldapuser10:/bin/bash
ldapuser11:*:1711:1711:LDAP Test User 11:/home/guests/ldapuser11:/bin/bash
...
```

3. As home directories, we probably want to use **rw** as the *mount option*.

Configuring indirect maps in **autofs** would look something like this:

```
# cat /etc/auto.master
/home/guests    /etc/auto.guests
# cat /etc/auto.guests
ldapuser1  -rw  instructor.example.com:/home/guests/ldapuser1
ldapuser2  -rw  instructor.example.com:/home/guests/ldapuser2
ldapuser3  -rw  instructor.example.com:/home/guests/ldapuser3
ldapuser4  -rw  instructor.example.com:/home/guests/ldapuser4
```

Each time a new LDAP user is created, **/etc/auto.guests** would need to be updated to include that additional user. However, notice the "pattern" to the lines. We want to support logging in as *any* username, so we could replace the first column with an "asterisk (\*)", a wildcard, matching any subdirectory name that the login process may try to **cd** to. Then, we use the metacharacter, "ampersand (&)", to replace the username in the share which carries over the mapname matched by the wildcard:

```
# cat /etc/auto.master  
/home/guests /etc/auto.guests  
# cat /etc/auto.guests  
* -rw instructor.example.com:/home/guests/&
```



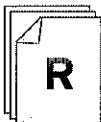
### Important

At the time of writing, there is a bug in the **autofs** init script (bugzilla #624444) that causes **service autofs restart** to sometimes fail. Until this is fixed, use one of the following:

**service autofs reload**

or

**service autofs stop && service autofs start.**



### References

Red Hat Enterprise Linux Storage Administration Guide

- Section 10.3: **autofs**

**autofs(5), auto.master(5)** man pages



## Practice Case Study

### Automounting NFS Directories

These steps should be performed on serverX.

Your company is now taking on several new clients:

1. The Organization of Secret Hidden Undertakings (or OSHU)
2. Race Along the Lake Investments, Inc. (or RALII)

Your company has a new NFS server for storing files related to these "special" clients named **instructor.example.com**, with two shares: **/var/nfs/oshu** and **/var/nfs/ralii**. The expectation is that more shares will be added as new clients are signed.

The workstations need to use **autofs** to automatically mount these shares to: **/special/oshu** and **/special/ralii**, respectively with read-only permissions.

Given the expectation that additional clients will be signed shortly, implement this using **autofs** wildcards and metacharacters.

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*

Test

## Criterion Test

Case Study

### Get Network User Information from an LDAP Directory Service

You will now configure serverX to get information about network users from an LDAP directory server available to all machines in the classroom.

Here is information that was provided to you about the LDAP server:

- Hostname: *instructor.example.com*
- Search Base DN: *dc=example,dc=com*
- CA Certificate: <http://instructor.example.com/pub/EXAMPLE-CA-CERT><sup>1</sup>

You will then configure serverX to automatically mount the home directories of your LDAP-based network users when they log in.

Here is information which was provided to you about the NFS storage server that contains the home directories:

- Hostname: *instructor.example.com*
- Exported Directory: **/home/guests/**

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*



## Personal Notes



## Unit Summary

### Network Authentication Using an LDAP Server

In this section you learned how to:

- Configure the system to authenticate users managed in a central LDAP directory service

### Network Mounting Home Directories

In this section you learned how to:

- Automount existing NFS home directories for remote users using indirect map metacharacters





## **UNIT ELEVEN**

# **CONTROLLING ACCESS TO FILES**

### **Introduction**

Topics covered in this unit:

- Managing Groups
- Access Control Lists

# Managing Groups

Like users, groups have a name and a number (GID). Local groups are defined in **/etc/group**. LDAP can be used to provide group information as well.

## Primary Groups

- Every user has exactly one *primary group*.
- For local users, the primary group is defined by the GID number of the group listed in the third field of **/etc/passwd**.
- Normally, the primary group owns new files which are created by the user.
- Normally, the primary group of a newly created user is a newly created group with the same name as the user. The user is the only member of this *User Private Group* (UPG).

## Supplementary Groups

- Users may be a member of zero or more *supplementary groups*.
- The users that are supplementary members of local groups are listed in the last field of the group's entry in **/etc/group**: For local groups, user membership is determined by a comma-separated list of users found in the last field of the group's entry in **/etc/group**:

```
groupname:password:GID:list,of,users,in,this,group
```

- Supplementary group membership is used to help ensure that users have access permissions to files and other resources on the system.



## Note

Given the automatic creation of User Private Groups (GID 500+), it is generally recommended to set aside a range of GID numbers to be used for Supplementary Groups. Here we will use the range of 200 to 400, though this does risk a collision with a System Group (GID 0-499).

## Managing Supplementary Groups

1. **groupadd -g 201 groupname** to create a supplementary group named **groupname** with a GID of **201**.
2. **usermod -aG groupname username** will add the user **username** to the group **groupname**.



### Important

The use of the **-a** option makes **usermod** function in "append" mode. Without it, the user would be removed from *all other supplementary groups*.



### References

Red Hat Enterprise Linux Deployment Guide

- Chapter 15: Users and Groups

**group(5), groupadd(8), groupdel(8), usermod(8) man pages**



### Practice Exercise

## Managing Groups

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

For the users that were created earlier on serverX, we will now put them into groups

<b>Group</b>	<b>groupname</b>	<b>user_list</b>
Professors	<i>profs</i>	<i>faraday, juliet, elvis</i>
Graduate Students	<i>grads</i>	<i>jack, kate, james, elvis</i>
Summer Interns	<i>interns</i>	<i>walt, ben, claire, hugo, elvis</i>

Table11.1. User and Group Assignments

Notice that there is one additional user that you will need to create named **elvis**, who should be placed in all three groups.

1. Create the groups specified in the table above, and assign the appropriate group members.
2. Use the **id** command to verify group memberships for all users.

# Managing File System Access Control Lists

## Access Control List Support

- Standard Linux file systems (ext2/3/4) support more complex file permissions to be set by using POSIX ACLs, provided the file system is mounted with the **acl** option.
- In Red Hat Enterprise Linux, if the last character of the permission string displayed by **ls -l** is a +, the file or directory has an ACL set.
- getfacl file** is used to display ACLs on a file

```
u:elvis:rwx      # applies to user elvis
u:3142:---      # applies to user id 3142
u::rwx          # applies to file's owner (user)

g:music:rwx     # applies to group music
g:10:r-x        # applies to group id 10
g::rw-          # applies to file's group

o::rwx          # applies to everyone else
```

- setfacl** is used to set or modify ACLs on a file

```
$ setfacl -m u:friend:r filename  # grants rw to user friend
$ setfacl -m g:grads:rw filename # grants rw to the group grads
$ setfacl -m g:prof:s:r filename # grants r to the group profs

$ setfacl -x u:friend           # removes the existing ACL for friend

$ setfacl -m o::- filename      # changes normal "other" permissions
```

Use this space for notes

## Permission Precedence

When determining whether a process (that is, a running program) can access a file, file permissions and ACLs are applied as follows:

- If the process is running as the user that owns the file, then the file's *user* permissions apply
- Else, if the process is running as a user that is listed in a user ACL entry, then the *user ACL* applies (as long as it is permitted by the **mask**)
- Else, if the process is running as a group that matches the group that owns the file or as a group with an explicit group ACL entry, if the permission is granted by *any* matching group it applies (as long as it is permitted by the **mask**)
- Otherwise, the file's *other* permissions apply

## The ACL Mask

- Files that have ACLs have a "mask" which limits the maximum permissions that both the group that owns the file, and that supplementary users and groups in ACLs, can have.
- **getfacl file** shows the current mask as **mask::permissions**.
- The group permissions displayed by **ls -ld file** also reflect the current mask (*not* the owning group's permissions!)



## Important

Changing group permissions on a file with an ACL by using **chmod** does not change the owning group's actual permissions! It actually changes the **mask**, which limits the maximum permissions of all groups and supplementary users that have access to the file. (That is, the permissions for everyone but the owning *user* and users in the *other* category for the file.)

Use this space for notes

---

## Default ACLs (Inheritance)

- Directories can have "default ACL" entries which are automatically set on new files created in that directory
- **setfacl -m d:u:elvis:rw directory** would set a default ACL entry granting read-write access to user **elvis** on all new files created in **directory**.
- This is similar to the way that the **setgid** permission, when set on a directory, causes new files created in that directory to be owned by the same group that owns the directory.



## Note

When setting default ACLs on a directory, if you want to ensure that users will be able to access the contents of new subdirectories created in it, make sure you include executable permissions on their ACL:

```
[user@host ~]$ setfacl -m d:u:elvis:rx directory
```

Normally, users will not automatically get executable on newly created regular files because unlike new directories, the ACL **mask** of a new regular file is **rw-** by default.

Use this space for notes

## ACL Mount Option

- Support for POSIX ACL entries must be enabled when the file system is mounted.
- The installer configures all **ext4** file systems it creates to automatically turn on ACL support.

```
[root@host ~]# dumpe2fs /dev/block-dev | grep 'Default mount'
Default mount options: user_xattr acl
```

- If you manually formatted the file system, you will need to mount it with the **acl** mount option.
- You can set a manually-formatted **ext4** file system to turn on support at mount automatically by using **tune2fs** to set default mount options:

```
[root@host ~]# tune2fs -o acl,user_xattr /dev/block-dev
```



### Note

The **user\_xattr** mount option does not have anything to do with POSIX ACLs, but enables *user extended attributes*. These are used by a handful of programs to store custom information with files. It is a good idea to leave this default option set even though it is not needed to enable POSIX ACLs. See **attr(5)** for more information.

Use this space for notes



## References

Red Hat Enterprise Linux Storage Administration Guide

- Chapter 16: Access Control Lists

**acl(5), getfac1(1), and setfac1(1) man pages**

## Collaborative Directory Permissions

In the quiz below, use both POSIX ACLs and standard permissions, as appropriate, to solve these problems.

1. Given a normal directory, where the owning *user* has **rwx** permissions, the owning *group* has **rwx** permissions, and *other* has **---** permissions, what command would grant a second group **r-x** permissions without changing the permissions of the existing owning group or *other*?
2. What command would automatically grant that second group read-write access to any newly created regular files in that directory?
3. *Bonus question.* What command would automatically set the owning *group* as the owning group of any newly created files in that directory?



Test

## Criterion Test

Case Study

### Using ACLs to Grant and Limit Access

This lab uses users and groups created earlier on serverX. If you do not already have the users and groups defined, run **lab-add-users** on serverX.

Graduate students need a collaborative directory titled **/opt/research**, where they can store generated research results. Only members of the groups **profs** and **grads** should be able to create new files in the directory, and new files should have the following properties:

- The directory should be owned by user **root**.
- New files should be group owned by the group **grads**.
- Professors (members of the group **profs**) should automatically have read/write access to new files.
- Summer interns (members of the group **interns**) should automatically have read-only access to new files.
- Other users (not a member of groups **profs**, **grads**, or **interns**) should not be able to access the directory and its contents at all.

See the Solutions appendix when you are done to check your solution and to see some possible approaches.

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*



## Personal Notes



## Unit Summary

### Managing Groups

In this section you learned how to:

- Identify primary versus supplementary groups
- Create a new group
- Delete a group
- Add a user account to a group

### Managing File System Access Control Lists

In this section you learned how to:

- Use an ACL entry to grant or block file access
- List the ACLs on a file
- Delete an ACL entry
- Assign an ACL or group ownership to new files created in a directory automatically





## **UNIT TWELVE**

# **MANAGING SELINUX**

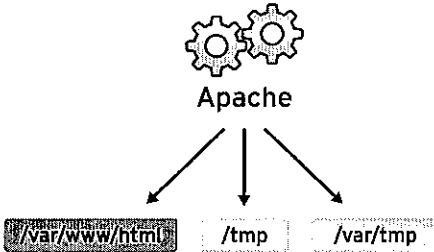
### **Introduction**

Topics covered in this unit:

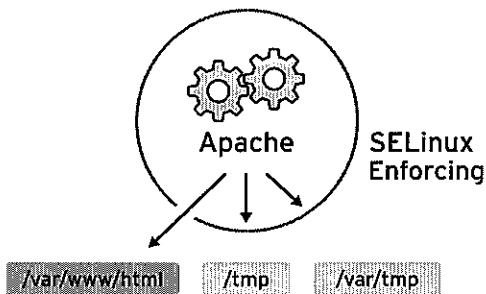
- Review basic SELinux concepts
- Displaying and setting SELinux modes
- Displaying and setting SELinux file contexts
- Tuning policy behavior with SELinux booleans
- Monitoring SELinux policy violations

## Basic SELinux Security Concepts

SELinux, Security-Enhanced Linux, is an additional method to protect your system.



Presuming we want to allow remote anonymous access to a web server, we must open the ports through the firewall. However, that means that malicious people can try to crack into the system through a security exploit and, if they compromise the web server process, gain its permissions: the permissions of the apache user and the apache group. That user/group has read access to things like the document root (`/var/www/html`), as well as write access to `/tmp`, `/var/tmp` and any other files/directories that are world writable.



SELinux is a set of security rules that determine which process can access which files, directories, ports, etc. Every file, process, directory and port has a special security label called SELinux contexts. A context is simply a name that is used by the SELinux policy to determine whether or not a process can access a file, directory or port. By default, the policy does not allow any interaction, so explicit rules grant access. If there is no allow rule, no access is allowed.

SELinux labels have several contexts, but we are most interested in the third context: the type context. Type context names usually end with `_t`. The type context for the web server is `httpd_t`. The type context for files and directories normally found in `/var/www/html` is `httpd_sys_content_t`. The type contexts for files and directories normally found in `/tmp` and `/var/tmp` is `tmp_t`. The type context for web server ports is `http_port_t`.

There is a rule in the policy that permits Apache (the web server process running as `httpd_t`) to access files and directories with a context normally found in `/var/www/html` and other web server directories (`httpd_sys_content_t`). There is no allow rule in the policy for files normally found in `/tmp` and `/var/tmp`, so access is not permitted. With SELinux, a malicious user could not access the `/tmp` directory, let alone write files to it. SELinux even has rules for remote filesystems such as NFS and CIFS, although all files on these filesystems are labeled with the same context.

One of the goals of SELinux is to protect the user data from system services that have been compromised.



## References

### Red Hat Enterprise Linux Security-Enhanced Linux

- Chapter 2: Introduction



Practice Quiz

## Basic SELinux Concepts

1. To which of the following does SELinux apply security context (check all that apply)?

*(select one or more of the following...)*

- a. Ports
- b. Processes
- c. Files
- d. Directories
- e. Remote file systems

2. SELinux can be used to:

*(select one or more of the following...)*

- a. Protect a service from running on other ports.
- b. Protect user data from applications like the web server
- c. Block remote systems from accessing local ports
- d. Keep the system updated
- e. Access a web server

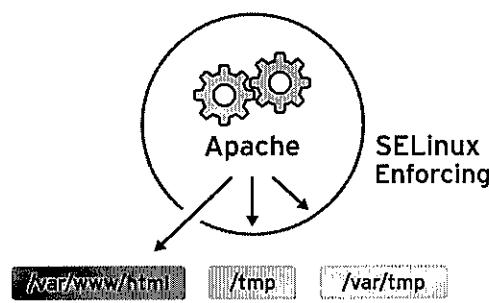
3. Which of the following are standard SELinux context types?

*(select one or more of the following...)*

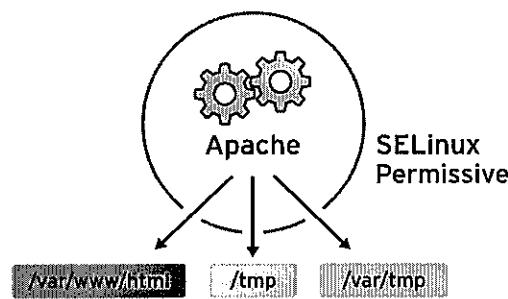
- a. selinux\_type
- b. object\_r
- c. httpd\_sys\_content\_t
- d. tmp\_t
- e. user\_u

## SELinux Modes

For troubleshooting purposes, we can temporarily disable SELinux protection, using SELinux modes.



In *enforcing mode*, SELinux actively denies access to the web server attempting to read files with `tmp_t` type context. In enforcing mode, SELinux both logs and protects.



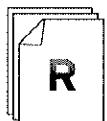
*Permissive mode* is often used to troubleshoot issues. In permissive mode, SELinux allows all interactions, even if there is no explicit rule, and it logs all of the denied interactions. This mode can be used to determine if you are having an SELinux issue. No reboot is required to go from enforcing to permissive or back again.

A third mode, *disabled*, completely disables SELinux. You must reboot to disable SELinux entirely, or to get from disabled mode to enforcing or permissive.



### Important

If you plan to re-enable SELinux restrictions, it is better to use permissive mode than to turn off SELinux entirely. One reason for this is that even in permissive mode, the kernel will automatically maintain SELinux file system labels as needed, avoiding the need for an expensive relabeling of the file system when you reboot with SELinux re-enabled.



## References

Red Hat Enterprise Linux Security-Enhanced Linux

- Section 5.5: SELinux Modes



Practice Quiz

## SELinux Modes

1. SELinux \_\_\_\_\_ mode allows logging, but not protection.

2. SELinux \_\_\_\_\_ mode protects the system.

3. Which of the following are valid SELinux modes?

*(select one or more of the following...)*

- a. enforcing
- b. testing
- c. permissive
- d. disabled
- e. logging

## Display and Modify SELinux Modes

Notice that **/etc/sysconfig/selinux** contains some useful comments:

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Use **/etc/sysconfig/selinux** to change the default SELinux mode at boot time. In the example above, it is set to enforcing mode.

To display the current SELinux mode, use **getenforce**. To modify the current SELinux mode, use **setenforce**:

```
[root@serverX ~]# getenforce
Enforcing
[root@serverX ~]# setenforce
usage: setenforce [ Enforcing | Permissive | 1 | 0 ]
[root@serverX ~]# setenforce 0
[root@serverX ~]# getenforce
Permissive
[root@serverX ~]# setenforce Enforcing
[root@serverX ~]# getenforce
Enforcing
```

### References

Red Hat Enterprise Linux Security-Enhanced Linux

- Section 5.5: SELinux Modes

**selinux(8), getenforce(1), setenforce(1)** man pages



Practice Exercise

## Changing Enforcing and Permissive Modes

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

1. On serverX, change the default SELinux mode to permissive and reboot.
2. After reboot, verify the system is in permissive mode.
3. Change the default SELinux mode to enforcing.
4. Change the current SELinux mode to enforcing.

## Display and Modify SELinux File Contexts

Many commands that deal with files have an option (usually **-Z**) to display or set SELinux contexts. For instance, **ps**, **ls**, **cp**, and **mkdir** all use the **-Z** option to display or set SELinux contexts.

```
[root@serverX ~]# ps axZ


| LABEL                         | PID | TTY | STAT | TIME | COMMAND       |
|-------------------------------|-----|-----|------|------|---------------|
| system_u:system_r:init_t:s0   | 1   | ?   | Ss   | 0:00 | /sbin/init    |
| system_u:system_r:kernel_t:s0 | 2   | ?   | S    | 0:00 | [kthreadd]    |
| system_u:system_r:kernel_t:s0 | 3   | ?   | S    | 0:00 | [migration/0] |
| ...                           |     |     |      |      |               |


[root@serverX ~]# service httpd start
[root@serverX ~]# ps -ZC httpd


| LABEL                            | PID   | TTY | TIME     | CMD   |
|----------------------------------|-------|-----|----------|-------|
| unconfined_u:system_r:httpd_t:s0 | 27672 | ?   | 00:00:00 | httpd |
| unconfined_u:system_r:httpd_t:s0 | 27675 | ?   | 00:00:00 | httpd |
| ...                              |       |     |          |       |


[root@serverX ~]# ls -Z /home
drwx-----. root      root      system_u:object_r:lost_found_t:s0 lost+found
drwx-----. student   student   unconfined_u:object_r:user_home_dir_t:s0 student
drwx-----. visitor   visitor   unconfined_u:object_r:user_home_dir_t:s0 visitor
[root@serverX ~]# ls -Z /var/www
drwxr-xr-x. root      root      system_u:object_r:httpd_sys_script_exec_t:s0 cgi-bin
drwxr-xr-x. root      root      system_u:object_r:httpd_sys_content_t:s0 error
drwxr-xr-x. root      root      system_u:object_r:httpd_sys_content_t:s0 html
drwxr-xr-x. root      root      system_u:object_r:httpd_sys_content_t:s0 icons

```

What determines a file's initial SELinux context? Normally, it is the parent directory. The context of the parent directory is assigned to the newly-created file. This works for commands like **vim**, **cp**, and **touch**, however, if a file is created elsewhere and the permissions are preserved (as with **mv** or **cp -a**), it will preserve the SELinux context as well. There are some special rules in the policy, called type transition rules, that may change the type context from the default. These rules are beyond the scope of this course.

```
[root@serverX ~]# ls -zd /var/www/html/
drwxr-xr-x. root      root      system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
[root@serverX ~]# touch /var/www/html/index.html
[root@serverX ~]# ls -Z /var/www/html/index.html
-rw-r--r--. root      root      unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/
index.html
```

**semanage fcontext** can be used to display or modify the rules that **restorecon** uses to set default file contexts. It uses extended regular expressions to specify the path and file names. The most common extended regular expression used in **fcontext** rules is **(/.\* )?** which means *optionally, match a / followed by any number of characters*. In essence, it will match the directory listed before the expression and everything in that directory recursively.

**restorecon** is part of the **policycoreutil** package, and **semanage** is part of the **policycoreutil-python** package.

```
[root@serverX ~]# touch /tmp/file1 /tmp/file2
[root@serverX ~]# ls -Z /tmp/file*
-rw-r--r--. root      root      unconfined_u:object_r:user_tmp_t:s0 /tmp/file1
-rw-r--r--. root      root      unconfined_u:object_r:user_tmp_t:s0 /tmp/file2
```

```
[root@serverX ~]# mv /tmp/file1 /var/www/html/
[root@serverX ~]# cp /tmp/file2 /var/www/html/
[root@serverX ~]# ls -Z /var/www/html/file*
-rw-r--r--. root root unconfined_u:object_r:user_tmp_t:s0 /var/www/html/file1
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
[root@serverX ~]# semanage fcontext -l
...
/var/www(/.*)?          all files
system_u:object_r:httpd_sys_content_t:s0

[root@serverX ~]# restorecon -Rv /var/www/
restorecon reset /var/www/html/file1 context unconfined_u:object_r:user_tmp_t:s0-
>system_u:object_r:httpd_sys_content_t:s0
[root@serverX ~]# ls -Z /var/www/html/file*
-rw-r--r--. root root system_u:object_r:httpd_sys_content_t:s0 /var/www/html/file1
-rw-r--r--. root root unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

The following example show how to use **semanage** to add a context for a new directory.

```
[root@serverX ~]# mkdir /virtual
[root@serverX ~]# touch /virtual/index.html
[root@serverX ~]# ls -Zd /virtual/
drwxr-xr-x. root root unconfined_u:object_r:default_t:s0 /virtual/
[root@serverX ~]# ls -Z /virtual/
-rw-r--r--. root root unconfined_u:object_r:default_t:s0 index.html
[root@serverX ~]# semanage fcontext -a -f "" -t httpd_sys_content_t '/virtual(/.*)?'
[root@serverX ~]# restorecon -RFvv /virtual
[root@serverX ~]# ls -Zd /virtual/
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 /virtual/
[root@serverX ~]# ls -Z /virtual/
-rw-r--r--. root root system_u:object_r:httpd_sys_content_t:s0 index.html
```

## References

Red Hat Enterprise Linux Security-Enhanced Linux

- Section 5.7: SELinux Contexts – Labeling Files

**restorecon(8)** and **semanage(8)** man pages





Practice Exercise

## Correcting SELinux File Contexts

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

You have been asked to adjust your remote machine's DNS configuration to exactly match the configuration from your desktop machine. You decide the easiest way is to copy the file **/etc/resolv.conf** from the local machine to the remote machine.

1. Transfer the **/etc/resolv.conf** file from your desktop machine to **root**'s home directory on serverX.
2. Shell into serverX as **root**. All of the following steps should occur on your server.
3. Observe the SELinux context of the initial **/etc/resolv.conf**.

Original **/etc/resolv.conf** context:

---

4. Move **resolv.conf** from **root**'s home directory to **/etc/resolv.conf**.
5. Observe the SELinux context of the newly copied **/etc/resolv.conf**.

New **/etc/resolv.conf** context:

---

6. Restore the SELinux context of newly positioned **/etc/resolv.conf**.
7. Observe the SELinux context of the restored **/etc/resolv.conf**.

Restored **/etc/resolv.conf** context:

---

# Managing SELinux Booleans

SELinux booleans are switches that change the behavior of the SELinux policy. SELinux booleans are rules that can be enabled or disabled. They can be used by security administrators to tune the policy to make selective adjustments. Many packages have man pages **\*\_selinux(8)** which may detail some of the booleans which they use; **man -k '\_selinux'** can find these man pages easily.

**getsebool** is used to display the booleans and **setsebool** is used to modify the booleans. **setsebool -P** modifies the SELinux policy to make the modification persistent. **semanage boolean -l** will show whether or not a boolean is persistent.

```
[root@serverX ~]# getsebool -a
abrt_anon_write --> off
allow_console_login --> on
allow_corosync_rw_tmpfs --> off
...
[root@serverX ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> off
[root@serverX ~]# setsebool httpd_enable_homedirs on
[root@serverX ~]# semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs --> off    Allow httpd to read home directories
[root@serverX ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
[root@serverX ~]# setsebool -P httpd_enable_homedirs on
[root@serverX ~]# semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs --> on    Allow httpd to read home directories
```

## References

Red Hat Enterprise Linux Security-Enhanced Linux

- Section 5.6: Booleans

**booleans(8), getsebool(8), setsebool(8), semanage(8) man pages**

## Monitoring SELinux Violations

The **setroubleshoot-server** package must be installed to send SELinux messages to **/var/log/messages**. **setroubleshoot-server** listens for audit messages in **/var/log/audit/audit.log** and sends a short summary to **/var/log/messages**. This summary includes unique identifiers (**UUIDs**) for SELinux violations that can be used to gather further information. **sealert -l *UUID*** is used to produce a report for a specific incident. **sealert -a /var/log/audit/audit.log** is used to produce reports for all incidents in that file.

```
[root@serverX ~]# touch /root/file3
[root@serverX ~]# mv /root/file3 /var/www/html
[root@serverX ~]# service httpd start
[root@serverX ~]# elinks -dump http://serverX/file3
                                         Forbidden
You don't have permission to access /file3 on this server.
[root@serverX ~]# tail /var/log/audit/audit.log
...
type=AVC msg=audit(1292526292.144:952): avc: denied { getattr } for
pid=27675 comm="httpd" path="/var/www/html/file3" dev=dm-1 ino=54545
scontext=unconfined_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:admin_home_t:s0
tclass=file
...
[root@serverX ~]# tail /var/log/messages
...
Dec 16 14:04:59 serverX setroubleshoot: SELinux is preventing /usr/sbin/httpd "getattr"
access to /var/www/html/file3. For complete SELinux messages. run sealert -l e6e1d1d6-
d716-4e2e-863c-bba4d2b2407a
[root@serverX ~]# sealert -l e6e1d1d6-d716-4e2e-863c-bba4d2b2407a
Summary:
```

SELinux is preventing /usr/sbin/httpd "getattr" access to /var/www/html/file3.

### Detailed Description:

SELinux denied access requested by httpd. /var/www/html/file3 may be a mislabeled. /var/www/html/file3 default SELinux type is **httpd\_sys\_content\_t**, but its current type is **admin\_home\_t**. Changing this file back to the default type, may fix your problem.

### Allowing Access:

You can restore the default system context to this file by executing the **restorecon** command. **restorecon '/var/www/html/file3'**, if this file is a directory, you can recursively restore using **restorecon -R '/var/www/html/file3'**.

### Fix Command:

```
/sbin/restorecon '/var/www/html/file3'
```



### Note

The "Allowing Access" section suggests `restorecon /var/www/html/file3`. If there may be other files that need to be adjusted, `restorecon` can recursively reset the context: `restorecon -R /var/www/`.



### References

Red Hat Enterprise Linux Security-Enhanced Linux

- Chapter 8: Troubleshooting
- **`sealert(8)` man page**



Practice Quiz

## Monitoring SELinux Violations

1. What file contains log entries providing unique identifiers for SELinux violations?  
\_\_\_\_\_
2. Given the UUID of an SELinux violation, what command generates a text report of the problem?  
\_\_\_\_\_

Test

## Criterion Test

Exercise

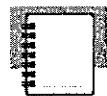
### Managing SELinux

#### ***Before you begin...***

Before you begin, run the **lab-setup-selinux** command on desktopX

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

1. Login to serverX as **student**. Open a terminal and switch to the **root** user.
2. Copy the **web\_content.tgz** archive from *instructor:/var/ftp/pub/materials* to **/tmp**.
3. Extract the archive into **/tmp**.
4. Move the extracted directory to **/var/www/html**.
5. Start the web service.
6. Try to observe the new directory with your web browser by visiting the URL [http://serverX/web\\_content](http://serverX/web_content).
7. Search your system for the UUIDs of any SELinux violations your attempt to browse the newly installed content might have generated.
8. Generate text reports for the violations.
9. Follow the report's advice to restore the SELinux contexts of the newly installed content.
10. Confirm that you can view the material from your web browser by visiting the URL [http://serverX/web\\_content](http://serverX/web_content).



## Personal Notes



## Unit Summary

### Basic SELinux Security Concepts

In this section you learned how to:

- Identify basic SELinux security concepts such as context, user/role/type, and policy

### SELinux Modes

In this section you learned how to:

- Describe the functional differences between SELinux enforcing and permissive modes when SELinux security is enabled

### Display and Modify SELinux Modes

In this section you learned how to:

- View and change the current SELinux mode of a system
- Set the default SELinux mode of a system

### Display and Modify SELinux File Contexts

In this section you learned how to:

- View the SELinux security context of processes and files
- Set the SELinux security context of files in the policy
- Restore the SELinux security context of files

### Managing SELinux Booleans

In this section you learned how to:

- Use SELinux booleans to make adjustments to policy behavior

### Monitoring SELinux Violations

In this section you learned how to:

- Deploy SELinux log analysis tools





## **UNIT THIRTEEN**

# **INSTALLING AND MANAGING SOFTWARE**

### **Introduction**

Topics covered in this unit:

- Using yum
- Package groups
- Using rpm
- Third-party repositories

## Using yum

**yum** is a powerful command-line tool that can be used to more flexibly manage (install, update, remove, and query) software packages. Official Red Hat packages are normally downloaded from Red Hat Network (RHN). When you register your machine with RHN, **yum** is automatically configured to use it. You can also configure **yum** to get packages from third-party package repositories over the network.

### Basic yum Commands

1. **yum help** will display usage information
2. **yum list** displays installed and available packages
3. **yum search *KEYWORD*** lists packages by keywords
4. **yum info *PACKAGENAME*** gives detailed information about a package
5. **yum install *PACKAGENAME*** obtains and installs a software package, including any dependencies
6. **yum remove *PACKAGENAME*** removes an installed software package, including any supported packages
7. **yum update *PACKAGENAME*** obtains and installs a newer version of the software package, including any dependencies. Generally the process tries to preserve configuration files in place, but in some cases they may be renamed if the packager thinks the old one will not work after update. With no *PACKAGENAME* specified, it will install all relevant updates.
8. **yum provides *PATHNAME*** displays packages that match the pathname specified (which often include wildcard characters)

Use this space for notes

---

---

---

### Example yum Commands:

To search for packages that have "web server" in their description, summary, or package name:

```
[root@serverX ~]# yum search 'web server'
=====
Matched: web server =====
mod_auth_mysql.x86_64 : Basic authentication for the Apache web server using a
                        : MySQL database
webalizer.x86_64 : A flexible Web server log file analysis program
freeradius.x86_64 : High-performance and highly configurable free RADIUS server
hsqldb.x86_64 : Hsqldb Database Engine
htdig.x86_64 : ht://Dig - Web search engine
htdig-web.x86_64 : Scripts and HTML code needed for using ht://Dig as a web
                        : search engine
httpd.x86_64 : Apache HTTP Server
```

To get information on the Apache HTTP Server:

```
[root@serverX ~]# yum info httpd
Available Packages
Name        : httpd
Arch       : x86_64
Version    : 2.2.15
Release    : 5.el6
Size       : 811 k
Repo       : base
Summary    : Apache HTTP Server
URL        : http://httpd.apache.org/
License    : ASL 2.0
Description: The Apache HTTP Server is a powerful, efficient, and extensible
              web server.
```

To install, update and remove the **httpd** package:

```
[root@serverX ~]# yum install httpd
[root@serverX ~]# yum update httpd
[root@serverX ~]# yum remove httpd
```



## Warning

**yum remove** will remove the package(s) listed and *any package that requires the package(s) being removed* (and package(s) which require those packages, and so on). This can lead to unexpected removal of packages, so carefully check the list of packages to be removed.



## References

Red Hat Enterprise Linux Deployment Guide

- Chapter 1: Yum

**yum(1), yum.conf(5)** man pages



Practice Exercise

## Searching for and Installing Packages

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

Perform the following steps on serverX unless directed otherwise.

1. Attempt to run the command **gnuplot**. You should find that it is not installed.
2. Search for plotting packages.
3. Find out more information about the **gnuplot** package.
4. Install the **gnuplot** package.
5. Attempt to remove the **gnuplot** package, but say no.

How many packages would be removed? \_\_\_\_\_

6. Attempt to remove the **gnuplot-common** package, but say no.

How many packages would be removed? \_\_\_\_\_

# Managing YUM Component Groups

**yum** also has the concept of *component groups*, which are collections of related software grouped around a particular solution. Review the output of the **yum help** command, filtering on lines with the phrase "group".

## Reading Activity: YUM Component Groups

Review the **yum(1)** man page and read the sections that describe the yum commands which start with "group".

1. **yum grouplist**

Lists all available groups

2. **yum groupinfo**

3. **yum groupinstall**

4. **yum grouperase**

5. **yum groupupdate**



### Note

The **yum groupupdate** command can be particularly useful after adding a new repository if the repository adds any packages to standard groups you have installed.



### References

**yum(1) man page**



**Practice Exercise**

## Managing YUM Component Groups

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

In this exercise you will gather information about the “PostgreSQL Database server” component group, and then you will install it on serverX

1. List all available component groups.
2. Find out more information about the *PostgreSQL Database Server* component group, including a list of included packages.
3. Install the *PostgreSQL Database Server* component group.

# Handling Third-Party Software

The **rpm** utility is a low-level tool that can get information about the contents of package files and installed packages. It gets its information from a local database or the package files themselves.

The general form of a query is:

- **rpm -q [select-options] [query-options]**
- **rpm --query [select-options] [query-options]**

## RPM Queries: Select Options

- **-q -a** - all installed packages
- **-q *PACKAGENAME*** - currently installed *PACKAGENAME*
- **-q -p *PACKAGEFILE.rpm*** - package file named *PACKAGEFILE.rpm*
- **-q -f *FILENAME*** - what package provides *FILENAME*

## RPM Queries: Information About Content of Packages

- **-q** - lists the package's name and version; compare to **yum list**
- **-q -i** - package information; compare to **yum info**
- **-q -l** - list of files installed by the specified package
- **-q -c** - list just the configuration files
- **-q -d** - list just the documentation files
- **-q --scripts** - list shell scripts that may run after the package is installed or uninstalled
- **-q --changelog** - list change information for the package



### Note

The **repoquery** command can also be used to get information about packages and their contents. It differs from **rpm** by looking up that information in yum's repositories and RHN instead of the local database of installed packages.

## Example rpm Query Commands:

Querying installed packages:

```
[root@serverX ~]# rpm -q samba-client  
samba-client-3.5.4-68.el6.x86_64  
[root@serverX ~]# rpm -q zlib -l
```

```
/lib64/libz.so.1
/lib64/libz.so.1.2.3
/usr/share/doc/zlib-1.2.3
/usr/share/doc/zlib-1.2.3/ChangeLog
/usr/share/doc/zlib-1.2.3/FAQ
/usr/share/doc/zlib-1.2.3/README
[root@serverX ~]# rpm -q httpd --scripts
preinstall scriptlet (using /bin/sh):
# Add the "apache" user
getent group apache >/dev/null || groupadd -g 48 -r apache
getent passwd apache >/dev/null || \
    useradd -r -u 48 -g apache -s /sbin/nologin \
        -d /var/www -c "Apache" apache
exit 0
postinstall scriptlet (using /bin/sh):
# Register the httpd service
/sbin/chkconfig --add httpd
preuninstall scriptlet (using /bin/sh):
if [ $1 = 0 ]; then
    /sbin/service httpd stop > /dev/null 2>&1
    /sbin/chkconfig --del httpd
fi
posttrans scriptlet (using /bin/sh):
/sbin/service httpd condrestart >/dev/null 2>&1 || :
```

Querying and installing package files:

```
[root@serverX ~]# cd /net/instructor/var/ftp/pub/materials/
[root@serverX ~]# rpm -q -p wonderwidgets-1.0-4.x86_64.rpm -l
/etc/wonderwidgets.conf
/usr/bin/wonderwidgets
/usr/share/doc/wonderwidgets-1.0
/usr/share/doc/wonderwidgets-1.0/README.txt
[root@serverX ~]# rpm -q -p wonderwidgets-1.0-4.x86_64.rpm -i
Name      : wonderwidgets                    Relocations: (not relocatable)
Version   : 1.0                                Vendor: Red Hat, Inc.
Release   : 4                                  Build Date: Fri 03 Dec 2010 05:42:55 AM EST
Install Date: (not installed)                 Build Host: station166.rosemont.lan
Group     : GLS/Applications                  Source RPM: wonderwidgets-1.0-4.src.rpm
Size      : 4849                               License: GPL
Signature : (none)
Summary   : Demonstration package for use in GLS training.
Description :
A demonstration package that installs an executable, and a config file.
[root@serverX ~]# rpm -q -p wonderwidgets-1.0-4.x86_64.rpm -c
/etc/wonderwidgets.conf
[root@serverX ~]# rpm -q -p wonderwidgets-1.0-4.x86_64.rpm -d
/usr/share/doc/wonderwidgets-1.0/README.txt
[root@serverX ~]# yum localinstall wonderwidgets-1.0-4.x86_64.rpm
...
Package wonderwidgets-1.0-4.x86_64.rpm is not signed
[root@serverX ~]# yum localinstall --nogpgcheck wonderwidgets-1.0-4.x86_64.rpm
[root@serverX ~]# rpm -q wonderwidgets
wonderwidgets-1.0-4.x86_64
```

## Using yum to Install Local Package Files

**yum localinstall PACKAGEFILE.rpm** can be used to install package files directly. It automatically downloads any dependencies the package has from RHN and any configured **yum** repositories. Packages are normally digitally signed to ensure they are legitimate; if the package

is not signed by a key trusted by your system, it will be rejected. The `--nogpgcheck` option can disable the signature check if you are certain the package is legitimate.



### Note

`rpm -ivh PACKAGEFILE.rpm` can also be used to install package files. However, using `yum` helps maintain a transaction history kept by `yum` (see `yum history`).



### Important

Be careful when installing packages from third parties, not just because of the software that they may install, but because the RPM may run arbitrary scripts as `root` as part of the installation process.

RPM package files are essentially `cpio` archives. The `cpio` command is an archiving tool like `zip` or `tar`, but differs in that it reads the list of files to be archived from STDIN rather than as command line arguments. Pipe the output of `rpm2cpio PACKAGEFILE.rpm` into `cpio -id`, and it will extract all the files stored in the RPM to the current directory.



### References

Red Hat Enterprise Linux Deployment Guide

- Section 3.2: Using RPM

`rpm(8)`, `repoquery(1)`, and `rpm2cpio(8)` man pages



**Practice Exercise**

## Handling Third-Party Software

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

In this exercise you will gather information about a third-party package, extract files from it, and install it as a whole on your desktopX system.

1. Download `wonderwidgets-1.0-4.x86_64.rpm` from <http://instructor/pub/materials>.
2. What files does it contain?
3. What scripts does it contain?
4. How much disk space will it use when installed?
5. Use `yum localinstall` to install the package.

# Using Third-Party yum Repositories

Third-party repositories are network-accessible directories of software package files which can be accessed by **yum**, provided outside of Red Hat Network. Yum repositories are used by non-Red Hat distributors of software, or for small collections of local packages. (For example, Adobe provides some of its free software for Linux through a yum repository.) The **instructor** classroom server actually hosts yum repositories for this class.

Put a file in the **/etc/yum.repos.d/** directory to enable support for a new third-party repository. Repository configuration files must end in **.repo**. The repository definition contains the URL of the repository, a name, whether to use GPG to check the package signatures, and if so the URL pointing to the trusted GPG key.

## Examples of **/etc/yum.repos.d/\*.repo** Configuration Files:

An example with a single repository, with security checks of downloaded packages disabled:

```
[GLS]
name=Instructor GLS Repository
baseurl=ftp://instructor.example.com/pub/gls
gpgcheck=0
```

An example with multiple repository references in a single file:

```
[base]
name=Instructor Server Repository
baseurl=http://instructor.example.com/pub/rhel6/dvd
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

# Optional rhel6
[optional]
name=Instructor Optional Repository
baseurl=http://instructor.example.com/pub/rhel6/Optional
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

[client]
name=Instructor Client Repository
baseurl=http://instructor.example.com/pub/rhel6/Client
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
enabled=0

[kernel-extras]
name=Instructor Kernel Extras Repository
baseurl=http://instructor.example.com/pub/rhel6/Kernel-Extras
gpgcheck=1
```



## Note

Note that some repositories, such as EPEL (Extra Packages for Enterprise Linux), provide this configuration file as part of an RPM package that can be downloaded from the web and installed with **yum localinstall**.

Installing the Red Hat Enterprise Linux 6 EPEL repo package:

```
[root@serverX ~]# rpm --import http://download.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-6
[root@serverX ~]# yum install http://download.fedoraproject.org/pub/epel/beta/6/x86_64/epel-release-6-5.noarch.rpm
[root@serverX ~]# cat /etc/yum.repos.d/epel.repo

[epel]
name=Extra Packages for Enterprise Linux 6 - $basearch
#baseurl=http://download.fedoraproject.org/pub/epel/6/$basearch
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-6&arch=$basearch
failovermethod=priority
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6

[epel-debuginfo]
name=Extra Packages for Enterprise Linux 6 - $basearch - Debug
#baseurl=http://download.fedoraproject.org/pub/epel/6/$basearch/debug
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-debug-6&arch=$basearch
failovermethod=priority
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6
gpgcheck=1

[epel-source]
name=Extra Packages for Enterprise Linux 6 - $basearch - Source
#baseurl=http://download.fedoraproject.org/pub/epel/6/SRPMS
mirrorlist=https://mirrors.fedoraproject.org/metalink?repo=epel-source-6&arch=$basearch
failovermethod=priority
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6
gpgcheck=1
```



## Important

Install the RPM GPG key before installing signed packages. This will verify that the package belongs to a key you have imported. Otherwise, **yum** will complain about the missing key. (You can use the **--nogpgcheck** option to ignore missing GPG keys, but this could cause forged or insecure packages to be installed on your system.)



## References

Red Hat Enterprise Linux Deployment Guide

- Section 1.3: Configuring Yum and Yum Repositories

**yum(1) and yum.conf(5) man pages**



Practice Exercise

## Using yum Repositories

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

You will configure your server to use a separate **yum** repository to obtain updates, and update your machine.

Perform the following steps on serverX unless directed otherwise.

1. Create the file **/etc/yum.repos.d/errata.repo**, to enable the "Updates" repository found on the instructor machine. It should access content found at the following URL: `ftp://instructor.example.com/pub/rhel6/Errata`
2. Update all relevant software provided by the repository, using **yum update**.

Test

## Criterion Test

Case Study

### Update and Install Software

***Before you begin...***

Run **lab-setup-server** on desktopX to prepare serverX for the exercise.

You have a new server, serverX, to administrate that has very specific software requirements. It must have the latest version of the following packages installed:

**kernel** (existing package w/ an update)

**xsane-gimp** (new package)

**bzip2** (updated package)

For security reasons it should not have the **xinetd** package installed.

Do not install all updates. Only install updates for the packages listed above if they are available.

Updated packages can be found at the following URL: <ftp://instructor.example.com/pub/rhel6/Errata>

When you are ready to check your work, run **lab-grade-packages-2** on serverX.

***How would you address the case study described above? Take notes on your process in the space below and then implement it.***



## Personal Notes



## Unit Summary

### Using yum

In this section you learned how to:

- List packages by name, keyword, or file
- Get the version and description of a package
- Install, update and remove packages with **yum**

### Managing YUM Component Groups

In this section you learned how to:

- List available package groups
- Display the packages in a package group
- Install a package group using **yum**.

### Handling Third-Party Software

In this section you learned how to:

- Query third-party packages for files before installation
- Query rpm package internals

### Using Third-Party yum Repositories

In this section you learned how to:

- Manage repository definition files in **/etc/yum.repos.d/**





## **UNIT FOURTEEN**

# **MANAGING INSTALLED SERVICES**

### **Introduction**

Topics covered in this unit:

- Manage Service Startup
- Verify Service

## Manage Services

*Daemons* are processes that wait or run in the background performing various tasks. Generally, daemons start automatically at boot time and continue to run until shutdown or they are manually stopped. By convention, the names of many daemon programs end in the letter "d".

Daemons are managed by *service scripts*, which are kept in the `/etc/rc.d/init.d/` directory. Service scripts are usually called with a single `start`, `stop`, `restart`, `status`, or `reload` argument. The easiest way to run service scripts is to use the `service` command.

Services are *enabled* (configured to start automatically at boot time) with `chkconfig service on`. Services are *disabled* (configured not to start automatically at boot time) with `chkconfig service off`. More sophisticated control over whether a service is on or off in particular runlevels is also possible.

Without arguments, `chkconfig` lists the current configuration of all services. The runlevels in which a service is enabled can be observed with `chkconfig --list service`. Running the `chkconfig` command does not immediately affect the state of a daemon started from a service script.



### Note

*Daemons* are background processes that do things. A *service script* may start one or more daemons, but service scripts may instead make a one time change to the state of the system (for example, to configure network interfaces) which does not involve leaving a daemon process running afterward. The `network` and `iptables` scripts are examples of this second kind of service script.

## Installing New Services

What command is commonly used to perform each of these steps of deploying a new service on a Red Hat Enterprise Linux system?

- **Install** the software: \_\_\_\_\_
- **Start** the service: \_\_\_\_\_
- **Enable** the service at bootup: \_\_\_\_\_
- **Test** the services: This will be covered on the next few pages.



## References

Red Hat Enterprise Linux Deployment Guide

- Chapter 7: Controlling Access to Services

**service(8) and chkconfig(8) man pages**

http://www.fedoraproject.org/wiki/Deployment\_Guide#Controlling\_Services

## Confirm Service Availability

### Diagnosing Network Service Problems

Perform the following steps on serverX unless directed otherwise.

```
[root@serverX ~]# service httpd restart
```

We have just restarted our **httpd** service. How can we confirm that it is working?

Assuming the restart fails, the following are checks that you can make to try to diagnose the problem. What are the commands that would show the following:

1. Is the daemon running?

If the daemon is not running, what likely next steps could you take?

- a. \_\_\_\_\_
- b. \_\_\_\_\_

2. Has the daemon bound to the correct ports?

If the daemon is running, but not listening to the correct port, what likely next steps could you take?

- a. \_\_\_\_\_
- b. \_\_\_\_\_

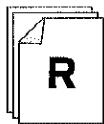
3. Can an external client connect to the port?

If the daemon is running and bound to the correct port, but outside clients cannot connect what is the likely candidate?

- a. \_\_\_\_\_

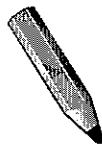
Use this space for notes

---



## References

**nc(1) and lsof(8) man pages.**



Test

## Criterion Test

Exercise

### Deploying an FTP Server

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

Install, start, enable and test the FTP service.

Perform the following steps on serverX unless directed otherwise.

1. Install the FTP server software, which is distributed as the `vsftpd` package.
2. Start the associated service.
3. Configure the service to start automatically at startup.
4. From your workstation, use the `lftp` client to connect to your server and verify that you can receive data from it.
5. Even if your service is successfully running, answer the following troubleshooting questions to confirm that the service has successfully deployed.
  - Is the daemon running?
  - Has the daemon bound to the correct port (TCP port 21)?
  - Can an external client connect to the port?



## Personal Notes



## Unit Summary

### Manage Services

In this section you learned how to:

- Start and stop services
- Enable and disable automatic starting of services
- List which services start automatically at boot time

### Confirm Service Availability

In this section you learned how to:

- Verify connectivity to a service
- Troubleshoot service availability



## **UNIT FIFTEEN**

# **ANALYZING AND STORING LOGS**

### **Introduction**

Topics covered in this unit:

- System Log Destinations
- Log Summary Reports
- Redirect Log Reports

## Determine Log Destinations

Many programs use a standard protocol to send messages to `rsyslogd`. Each message is described by a *facility* (the type of message it is) and a *severity* (how important it is). The names of available facilities and levels of severity are standardized. The `/etc/rsyslog.conf` file uses the facility and severity of a log message to determine where it should be sent (to a log file, for example).

The `rsyslog.conf` file is documented by the `rsyslog.conf(5)` man page and by extensive HTML documentation in `/usr/share/doc/rsyslog-*/manual.html`. The ##### RULES ##### section of `/etc/rsyslog.conf` contains directives that define where log messages are saved. The left-hand side of each line indicates the facility and severity of the log message the directive matches. The right-hand side of each line indicates what file to save the log message in. Note that log messages are normally saved in files in the `/var/log` directory.

### Sample `rsyslog.conf` file

```
##### MODULES #####
$ModLoad imuxsock.so      # provides support for local system logging (e.g. via logger
                           command)
$ModLoad imklog.so        # provides kernel logging support (previously done by rklogd)
#$ModLoad immark.so       # provides --MARK-- message capability

# Provides UDP syslog reception
#$ModLoad imudp.so
#$UDPServerRun 514

# Provides TCP syslog reception
#$ModLoad imtcp.so
#$InputTCPServerRun 514

##### GLOBAL DIRECTIVES #####
# Use default timestamp format
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat

##### RULES #####
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                                     /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none    /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                    /var/log/secure

# Log all the mail messages in one place.
mail.*                                         -/var/log/maillog

# Log cron stuff
cron.*                                         /var/log/cron
```

```
# Everybody gets emergency messages  
*.* emerg  
  
# Save news errors of level crit and higher in a special file.  
# uucp,news.crit  
# /var/log/spooler  
  
# Save boot messages also to boot.log  
local7.* /var/log/boot.log
```

## Default Log Files

Fill in the name of the log file as you review the contents of **/etc/rsyslog.conf** on serverX.

1. All authentication-related messages go to \_\_\_\_\_

\_\_\_\_\_

2. Anything e-mail related goes to \_\_\_\_\_

\_\_\_\_\_

3. Messages related to cron go to \_\_\_\_\_

\_\_\_\_\_

4. All other messages sent at **info** priority or higher are saved in \_\_\_\_\_

\_\_\_\_\_

## Rotating Logs

- Logs are "rotated" to keep them from filling up the file system containing **/var/log/**.
- When a log file is rotated, it is renamed with an extension indicating the date on which it was rotated: the old **/var/log/messages** file may become **/var/log/messages-20101030** if it is rotated on October 30, 2010.
- Once the old log file is rotated, a new log file is created and the service that writes to it is notified.
- After a certain number of rotations (typically after four weeks), the old log file is discarded to conserve disk space.
- A cron job runs the logrotate program daily to see if any logs need to be rotated.
- Most log files are rotated weekly, but logrotate rotates some faster, or slower, or when they reach a certain size.



## References

**rsyslog** Manual

- [/usr/share/doc/rsyslog-\\*/\\*manual.html](/usr/share/doc/rsyslog-*/*manual.html)

**rsyslog.conf(5)** and **logrotate(8)** man pages



---

### Practice Quiz

## Review rsyslog

Answers the following questions:

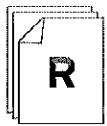
1. Which two fields are used to match log events?
2. What is the effect of a wildcard in the first field?
3. What is the effect of a wildcard in the second field?
4. Is it possible for the same log event to be recorded in more than one log?

## Locate and Analyze a Log Summary Report

A program called **logwatch** can be installed which will automatically analyze the standard log files and send a summary e-mail to **root**. A quick way to check the status of a system is to read this log summary report.

**logwatch** runs as a daily cron job to generate its report of interesting log information. This report is e-mailed to the local **root** account, by default.

If you want to generate a logwatch email manually, simply run **logwatch**.



### References

`/usr/share/doc/logwatch-*`

**logwatch(8)** man page



#### Practice Performance Checklist

## Analyze a Log Summary Report

Determine the amount of free space for the root filesystem from the latest logwatch report on serverX.

- Open the email of root
- Locate and read the most recent logwatch report. If there is no logwatch report, run the **logwatch** command to generate one manually.
- Record the amount of free space for the / filesystem:

## Change the Log Summary e-mail Address

**/etc/logwatch/conf/logwatch.conf** is an empty file which contains local logwatch settings. The system wide default settings for **logwatch** are kept in **/usr/share/logwatch/default.conf/logwatch.conf**. An excerpt is shown below:

```
# Default Log Directory
# All log-files are assumed to be given relative to this directory.
LogDir = /var/log

# You can override the default temp directory (/tmp) here
TmpDir = /var/cache/logwatch

# Default person to mail reports to. Can be a local account or a
# complete email address. Variable Print should be set to No to
# enable mail feature.
MailTo = root

# Default person to mail reports from. Can be a local account or a
# complete email address.
MailFrom = Logwatch

# if set, the results will be saved in <filename> instead of mailed
# or displayed.
#Save = /tmp/logwatch

# The default time range for the report...
# The current choices are All, Today, Yesterday
Range = yesterday

# The default detail level for the report.
# This can either be Low, Med, High or a number.
# Low = 0
# Med = 5
# High = 10
Detail = Low

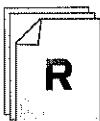
# The 'Service' option expects either the name of a filter
# (in /usr/share/logwatch/scripts/services/*) or 'All'.
# The default service(s) to report on. This should be left as All for
# most people.
Service = All

# By default we assume that all Unix systems have sendmail or a sendmail-like system.
# The mailer code Prints a header with To: From: and Subject:
mailer = "sendmail -t"

# By default the cron daemon generates daily logwatch report
# if you want to switch it off uncomment DailyReport tag.
# The implicit value is Yes
#
# DailyReport = No
```

Normally you edit the file in **/etc**, not the defaults in **/usr/share**. The **MailTo** element takes an e-mail address to send logwatch reports to; by default logwatch uses **root@localhost**. If you wanted **student@serverX.example.com** to receive these mailings, add the following line to the **/etc/logwatch/conf/logwatch.conf** file:

MailTo = student@serverX.example.com



## References

**logwatch(8) man page**



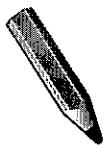
Practice Case Study

## Redirect Log Summary e-mails

Change the configuration of **logwatch** on serverX to send log summary reports to user **student** rather than user **root**.

*Bonus Question:* How would you send the **logwatch** reports to both **student** and **root**?

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*



Test

## Criterion Test

Case Study

### Log Debugging Messages

Your company wants to include a debug log on your server for analysis. Redirect all debugging level messages (and higher priority) to a file named `/var/log/debug.log`.

*How would you address the case study described above? Take notes on your process in the space below and then implement it.*



## Personal Notes



## Unit Summary

### Determine Log Destinations

In this section you learned how to:

- Manage the dispatching of software logs with `rsyslog`
- Protect filesystem with `logrotate`

### Locate and Analyze a Log Summary Report

In this section you learned how to:

- Utilize log summary emails

### Change the Log Summary e-mail Address

In this section you learned how to:

- Manage the dispatching of log summaries





## **UNIT SIXTEEN**

# **MANAGING PROCESSES**

### **Introduction**

Topics covered in this unit:

- Monitoring processes
- Terminating processes
- Schedule periodic tasks
- Deferring tasks

## Monitoring Processes

A **process** is an instance of a running program. The **ps** command can be used to list processes. By default, it gives you very little useful information. It is only showing us processes started from this terminal (and in an X session, every window is a terminal...). If asked for help, however, the **ps** command has more command line switches than you want to know about, and can be tailored to provide very concise information. Everyone has their favorite options, but here is one we suggest you use:

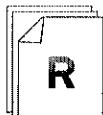
```
# ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root        1  0.0  0.0    4132   888 ?      Ss 17:44 0:01 /sbin/init
root        2  0.0  0.0      0     0 ?      S  17:44 0:00 [kthreadd]
root    1469  0.0  0.0   63572  1232 ?      Ss 17:45 0:00 /usr/sbin/sshd
rlocke  2348  0.0  0.0  107956  1852 pts/0    Ss 17:46 0:00 bash
rlocke  2711  0.0  0.1  150652  3628 pts/1    S+ 17:54 0:00 vim Processes.xml
rlocke  3507  2.0  0.0  107564  1032 pts/0    R+ 19:00 0:00 ps aux
...

```

The **top** utility displays an automatically updating list of current processes. The following keys can be used within **top** to manage the display:

- M: Sort processes by memory utilization
- P: Sort processes by processor utilization
- h: Display help about more commands
- q: Quit

When ordering processes by memory use, **top** considers **RES** more important than **VIRT**. **RES** is the amount of physical memory currently used by a process (the “resident set”). **VIRT** is the space in its virtual memory map currently reserved by the process for possible use, and is much less meaningful. Even **RES** can be an over-estimate, as it includes memory shared with other processes. In **ps**, **VIRT** is labeled **VSZ**, and **RES** is **RSS**.



### References

**ps(1)** and **top(1)** man pages

# Terminating and Governing Processes

Processes communicate using messages called *signals*. Signals can arrive at any time (asynchronously). Signals do not carry information beyond the *signal number*, which indicates what kind of signal it is.

Processes handle the signal in different ways depending on the signal number; the process can exit, exit and dump a copy of what is in its memory, ignore the signal, or do something else. Most often, when a user signals a process, it is to make it terminate (exit). System events can send signals, or a user can send an arbitrary signal to a process with the **kill** command.

## Signals

1. **top** and **kill** can both be used to send a signal to a process.
2. **kill -l**: Displays a table of the defined signal numbers.
3. **kill -9 3254**: Sends signal number 9 to the process with PID 3254.

Number	Name	Function
1	HUP	Reinitialize a daemon
9	KILL	Force a process to terminate immediately
15	TERM	Request a process to terminate after cleanup (DEFAULT)

Table 16.1. Good Signals to Know

In the table above, circle the signal that a programmer cannot override. (The **kill(1)** man page will point this out if your instructor does not.)

## Process Scheduling (Niceness)

- A Linux system can have as many processes running at the same time as the total number of logical processing units on its CPUs.
- The system appears to have more processes "running" by making them take turns on available logical processing units ("time slicing").
- By default, every process has equal access to CPU time.
- The *niceness* of a process can be changed to adjust the priority of the process, giving it a larger or smaller share of CPU time compared to other processes.
- Niceness is a value which ranges from -20 (very greedy), through the default of 0, to 19 (very nice to other processes).
- Users can increase the niceness of their processes (requesting a smaller share of time).
- Only root can decrease the niceness of processes (requesting a larger share of time).
- **renice** and **top** can both be used to change the niceness of a running process.
- **nice** can be used to set the niceness of a new process.



## Important

Niceness only affects the *share* of CPU time a particular process gets relative to other processes; if other processes are not running, a low priority process will still get all the CPU time on one logical processing unit.



## Workshop

### Using top to Manage Processes

*Follow along with the instructor as you complete the steps below.*

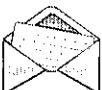
1. Open two terminal windows from a graphical session on serverX.
2. In the first terminal, run **top**.
3. In the second terminal, run:

```
cat /dev/zero > /dev/null &
```

4. Start four (4) more **cat** processes as above.
5. In **top**, note that the **cat** processes each have a roughly equal share of CPU time.
6. Using **top**, determine the PID of one **cat** process.
7. Use **renice** in the second terminal to adjust the niceness of the process to **10**:

```
renice -n 10 PID
```

8. Choose a different **cat** process, and make it greedy by using renice to change its priority to **-5**. Again observe the relative CPU utilization.



## Note

You will need to have **root** privileges to give it a higher priority (lower nice value).

9. Use **nice** to start a new **cat** process with a lower priority:

```
nice -n 5 cat /dev/zero > /dev/null &
```

10. Use the **r** key in **top** to renice some of the **cat** processes.
11. Use the **k** key in **top** to kill all of the **cat** processes.



## References

**kill(1), signal(7), nice(1), renice(1), and top(1) man pages**



Practice Exercise

## Managing Processes

***Before you begin...***

Run **lab-setup-processes** on serverX to prepare for the exercise.

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

1. Change the priority of the process that is using the most CPU resources to 5.
2. Terminate the process that is using the most memory resources.
3. When you are ready to check your work, run **lab-grade-processes** on serverX.
4. After you have successfully completed the exercise and checked your work, run **lab-cleanup-processes** on serverX to clean up.

# Managing Periodic Tasks

The cron facility manages programs that must be run on a repeating, periodic schedule. A daemon, **crond**, wakes up once a minute to run any tasks that have been scheduled. Users schedule personal tasks with the **crontab** command. The system administrator can set tasks in system-wide configuration files.

Individual users register tasks using a text file referred to as a crontab ("cron table") file. **crontab -l** lists the file. **crontab -r** deletes the file. **crontab -e** edits the file. The default editor used with **crontab -e** is **vi**.

The **crontab -e** command opens a blank crontab file. The syntax of a crontab file is documented in the **crontab(5)** man page. Each line defines either a scheduled task or an "environment variable" that affects task execution. Task lines have six fields: the first five fields define the minute, hour, day of the month, month, and day of the week, while the remainder of the line specifies a command to run.



## Important

The command runs when *either* day of month *or* day of week match! (It doesn't run just when both criteria match, which is the behavior most people expect.)

The following example shows a cron entry that runs the **ls** command every minute:

\* 1 \* 2 \* 3 \* 4 \* 5 ls

- ➊ Minute. Valid values are 0-59
- ➋ Hour. Valid values are 0-23. A 20 means 8 PM.
- ➌ Day of month. Valid values are 1-31
- ➍ Month. Valid values are 1-12 or the first three letters of the name (e.g., Jan)
- ➎ Day of week. Valid values are 0-7 (0 or 7 is Sunday) or the first three letters of the name (e.g., Sun)
- ➏ Command. Add the command as you would type it on the command-line. Use the semi-colon (;) to separate multiple commands to run at the same time. Add a separate line to run a command at a different time.



## Note

The wildcard character (\*) is used to match every value in the column. For instance, in the **ls** cron job above, the first character represents all minutes. Another way to specify this is: **0,1,2,3,4,5...,58,59 or 0-59**.

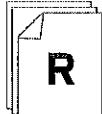
This can also be used to specify a *step*. For instance, **\*/5** in the first column would match every five values: **0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55**.

<b>Cron syntax</b>	<b>When the command will run</b>
<b>05 * * * *</b>	Every hour at 5 minutes past the hour
<b>05 02 * * *</b>	Every day at 2:05 am
<b>30 08 01 * *</b>	8:30 pm on the first of every month
<b>00 07 25 12 *</b>	December 25th at 7:00 am
<b>30 16 * * 5</b>	Every Friday at 4:30pm
<b>*/5 * * * *</b>	Every 5 minutes (0,5,10...45,50,55)
<b>*/10 9-16 1,15 * *</b>	Every 10 minutes between 9 am and 5 pm (it will not run at 5 pm--the last instance will run at 4:50pm) on the first and fifteenth of the month
<b>0 0 1 jan 0</b>	January 1 at midnight <i>and</i> every Sunday in January (not just if January 1 is on Sunday)

Table16.2. Example cron table entries

Tasks that belong to the system normally go in system crontab files instead of personal ones. The main crontab file is **/etc/crontab** which can be edited normally (**crontab -e** is not used). A better practice is to create a crontab file in a normal text editor and drop it into **/etc/cron.d/** (this avoids issues when the **cronie** package is updated; this is what system packages do). **/etc/cron.d/** crontab files have an extra field after the date specification indicating the user which will run the job.

Scripts which need to run once a day, week, or month can simply be set executable and dropped into the appropriate directory in **/etc/cron.{daily,weekly,monthly}**. These tasks are run by the system anacron service, configured by **/etc/anacrontab**. In Red Hat Enterprise Linux 6, **anacron** is an integrated component of cron to better manage these jobs and ensure that they are run after boot if missed because the machine is off. For more information see **anacrontab(5)** man page.



## References

**crontab(1)**, **crontab(5)**, **anacrontab(5)**, and **crontabs(4)** man pages



## Practice Quiz

## cronie Scheduling

1. When will the following jobs run?

a. `00 07 25 12 * /usr/local/bin/open_presents`

b. `*/5 * * * * /usr/local/bin/take_stats`

c. `07 03 * * * /sbin/service xend restart`

d. `30 16 * * 5 /usr/local/bin/mail_checks`

2. Devise a cron entry which would run the script `/usr/local/bin/vacuum_db` once a month on the first day of the month.
3. What if the machine in the previous question was down for maintenance on February first? What would be a better way to insure the database doesn't operate 2 (or more) months between vacuuming?

## Scheduling Deferred Tasks

### Reading Activity: The at Command

Using the **at** command, you can specify a task (referred to as a *job*) to run at a specific time in the future. The job might be a one-time backup, a check of your system, or a notification sent at a specific time. Jobs that might take a long time to complete also are good candidates for the **at** command. To do this, just use **at** to set the task to run a minute or two in the future. Then you can safely log off, since the task runs disconnected from your shell session.

The **at** command must specify when the task should run. That specification can be a specific time and/or date (**Monday, 10pm**, or **July 15**, for example). It can be relative to the current time (**now +5 min**, **now +3 days**, or **4pm +1 week**). By adding other options, you can have mail sent when the task completes (**-m**) or read the task from a file (**-f file**) instead of from standard input.

After typing the **at** command line press **Enter** and proceed to type in the commands to be included in the job. The task can consist of multiple commands. When you are done typing the commands to run, press **Ctrl+d** on a line by itself to complete the task. For example:

```
[user@host ~]$ at now +2 min
at> echo "Hello from the at command" > /dev/pts/2
at> Ctrl-D <EOT>
job 1 at 2011-01-28 00:38
[user@host ~]$
```

Once an **at** job is set to run, you can list job numbers and times using the **atq** command. To see the commands included in the job, type **at -c #** (replacing **#** with the job number you want to view). Before a job runs, you can remove it (as long as it is your job or you are root) by typing the **atrm #** command, where **#** is replaced by the job number.

### Instructions

Read the **at(1)** man page focusing on the **at**, **atq**, **atrm** commands, then answer the questions below.

1. What does the **at** command do?
  
2. What is the syntax of the **at** command?
  
3. How would you express the following time specifications to the **at** command?
  - To run a job at 4 pm three days from now?
  
  - To run a job at 10:00 am next July 31st?

- To run a job at 1 am, tomorrow?
4. What does the **atq** command do?
  5. What is the format of the output of the **atq** command?
  6. What does the **atrm** command do?
  7. What is the syntax of the **atrm** command?



## References

[at\(1\) man page](#)



Practice Performance Checklist

## Scheduling Deferred Tasks

Perform the following steps on serverX unless directed otherwise.

- Determine the current system time.
- Interactively schedule the **who** command to run 1 minute into the future.
- While waiting for the **at** command to complete, create a simple text file containing the command **cal**. Make it executable.
- Schedule the command created above to execute at teatime, both today and tomorrow.
- Switch places with a neighbor to perform the following two steps.
  - Use **atq** to confirm the pending at jobs.
  - Use **atrm** to remove the job for teatime tomorrow.
- Switch back, and check your mail to confirm that the original **who** command executed.



Test

## Criterion Test

Exercise

### Managing Processes

*Before you begin...*

Run `manage_processes_start` on serverX to prepare for the exercise.

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

Perform the following steps on serverX unless directed otherwise.

1. After running the `manage_processes_start` command, serverX should now be very sluggish.
2. Determine the process which is using excessive amounts of memory, and terminate the process.
3. Determine the process which is using excessive amounts of CPU, and renice it to a niceness of 15.
4. Create a system cron job which once every half hour readjusts all processes owned by the user `elvis` to a niceness of 10.

*Hint: use the `renice(1)` man page to determine the option to adjust all user processes.*

5. Register a job to execute once at 3 am which runs the command `ps aux`.



## Personal Notes



## Unit Summary

### Monitoring Processes

In this section you learned how to:

- Identify processes which consume the most CPU resources

### Terminating and Governing Processes

In this section you learned how to:

- Terminate processes
- Change the priority of an existing process
- Launch a process with a non-default priority

### Managing Periodic Tasks

In this section you learned how to:

- Schedule recurring jobs using **cron**

### Scheduling Deferred Tasks

In this section you learned how to:

- Schedule a one-time task using **at**
- List and delete pending one-time tasks





## **UNIT SEVENTEEN**

# **TUNING AND MAINTAINING THE KERNEL**

### **Introduction**

Topics covered in this unit:

- Supported Architectures
- Kernel Modules
- Kernel Parameters
- Kernel Upgrades

## Supported Architectures and Kernel Identification

As the interface between user programs and the system hardware, the kernel plays a key role in ensuring that Red Hat Enterprise Linux can be used on a wide variety of physical and virtual hardware environments.

### Red Hat Enterprise Linux 6 Supported Architectures

Red Hat Enterprise Linux 6 is supported to run directly on four processor architectures:

- Intel and AMD 64-bit x86-64
- Intel and AMD 32-bit x86
- IBM POWER (64-bit POWER6 or later)
- IBM System z (System z9 or later)

This class is taught on machines using the Intel and AMD 64-bit x86-64 architecture (also known as *AMD64* and *Intel 64*).

### Red Hat Enterprise Linux 6 and Virtualization

#### Red Hat Enterprise Linux 6 Supported as Virtualized Guest

Red Hat Enterprise Linux 6 is supported when running as a virtualized guest on the following hypervisors:

- KVM in Red Hat Enterprise Linux 5 and 6 (x86-64)
- Xen in Red Hat Enterprise Linux 5 (x86 and x86-64, paravirtualized and fully virtualized)
- VMware ESX Server and VMware ESXi Server
- Microsoft Windows Server 2008 Hyper-V

This means that Red Hat Enterprise Linux on these virtualization platforms is certified and supported independently of the underlying physical hardware (aside from any hardware pass-through). Please note that Red Hat supports the guest operating system. Direct support for a virtualization platform comes from the respective vendor. Please refer to <http://hardware.redhat.com/> for details on the verified products and versions, and to [http://www.redhat.com/rhel/server/virtualization\\_support.html](http://www.redhat.com/rhel/server/virtualization_support.html) for more information.

#### Red Hat Enterprise Linux 6 Support in Hardware Partitions

Red Hat Enterprise Linux is also supported on hardware partitioning and virtualization solutions such as

- IBM POWER and System z
- Fujitsu PRIMEQUEST

These are certified for support with the physical hardware, not as a generic feature on non-certified hardware. See <http://hardware.redhat.com/> for details.

## Red Hat Enterprise Linux 6 Support on the Public Cloud

For private in-house cloud computing deployments, standard support applies. Red Hat also certifies running Red Hat Enterprise Linux on the platform of various public cloud computing providers as part of our Cloud Partner Program. At the time of writing, Certified Cloud Providers included:

- Amazon EC2 (<http://www.redhat.com/solutions/cloud/amazon/>)
- IBM (<http://www.ibm.com/ibm/cloud/>)
- Savvis (<http://www.savvis.com/>)

## System Limits

Supported system limits depend on architecture and product variant and version implemented. The URL <http://www.redhat.com/rhel/compare> is updated as new versions are released and new hardware is qualified.

## Identifying the Running Kernel

1. `cat /etc/redhat-release` - installed Red Hat Enterprise Linux release
2. `uname -r` - Kernel version currently running
3. `yum list installed kernel\*` - installed kernel versions
4. `uname -m` or `arch` - processor architecture currently running on

Occasionally, the kernel emits log messages. These messages are logged in the `/var/log/messages` file, labeled as the `kernel` service.



## References

- Red Hat Server and Desktop Version Comparisons  
<http://www.redhat.com/rhel/compare/>
- Virtualization Support in Red Hat Enterprise Linux  
[http://www.redhat.com/rhel/server/virtualization\\_support.html](http://www.redhat.com/rhel/server/virtualization_support.html)
- Red Hat Hardware Catalog  
<http://hardware.redhat.com/>
- Cloud Partner Program  
<http://www.redhat.com/solutions/cloud/partners/>
- `uname(1)` and `arch(1)` man pages

# Managing Kernel Modules

*Kernel modules* are object files, executable code that can be dynamically linked into a Linux kernel while it is running to extend its capabilities or provide device drivers. Dynamically loadable kernel modules are useful because they allow Linux to load only components of the kernel that are needed on a particular system in a particular configuration, saving memory space and system resource usage. They also allow the kernel to be extended without the need to recompile it and reboot the system.

## Module Loading and Unloading

- The core kernel image, loaded at boot time, resides at **/boot/vmlinuz-VERSION**.
- While multiple kernels can be installed, only one is the current running kernel. In order to change kernels, the system has to be rebooted.
- Every kernel includes a collection of dynamically loaded modules compatible with that kernel, which are kept in **/lib/modules/VERSION/**.
- Generally, modules are loaded and unloaded on demand, with no user (or administrator) interactions.
- Currently loaded modules can be listed with **lsmod**.
- Occasionally, modules may need to be loaded manually with **modprobe MODULENAME**.
- Modules which are no longer in use can be removed with **modprobe -r MODULENAME**.

## Module Parameters

- Many modules accept parameters which can be specified as the module is loaded.
- The **modinfo** command reveals the parameters that a module supports.
- Parameters are specified as **name=value** pairs on the **modprobe** command line.

```
# modprobe ecryptfs ecryptfs_verbose=1
```

- Parameters can be applied automatically by configuring options in a **/etc/modprobe.d/local.conf** configuration file:

```
options ecryptfs ecryptfs_verbose=1
```

## References

- Red Hat Enterprise Linux Deployment Guide
  - Chapter 22: Working with Kernel Modules

**lsmod(8)** and **modprobe(8)** man pages



**Practice Exercise**

## Loading Modules and Setting Default Parameters

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

Some features of the kernel-based firewall are implemented as optional kernel modules, like performing connection tracking on an ftp server.

You have been asked to load the *nf\_conntrack\_ftp* kernel module, and configure it appropriately for an FTP server listening on TCP port 21 and on 8021.

The following commands are to be run on your serverX.

1. Use the locate command to convince yourself that the *nf\_conntrack\_ftp* module is supported by your kernel.
2. Load the FTP connection track module.
3. Convince yourself it's loaded.
4. Unload the FTP connection track module.
5. Convince yourself it's unloaded.
6. In addition to the standard port 21, you are planning to run an FTP server on the non-standard port 8021. Examine options which might let you specify non-standard ports.
7. Configure a file **/etc/modprobe.d/local.conf** which implements this option upon loading

# Specifying Kernel Boot Parameters

## Kernel Command Line Parameters

- The kernel can be configured by passing it command line parameters.
- The command line used to start your current kernel can be examined in `/proc/cmdline`.
- Many (but not all) kernel command line options are documented in the `bootparam(7)` man page. Additional parameters are documented in the `kernel-doc` package, in the file `/usr/share/doc/kernel-doc-2.6.32-*/Documentation/kernel-parameters.txt`.
- Command line options are specified in the bootloader's configuration file, `/boot/grub/grub.conf`.
- Anything not recognized by the kernel is passed on as an environment variable or argument to the first process.

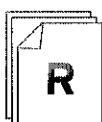


## Warning

Ensure that you understand the likely effects of a kernel parameter before setting it.

Use this space for notes

---



## References

- | Kernel documentation: "Kernel Parameters"  
| `/usr/share/doc/kernel-doc-2.6.32-*/Documentation/kernel-parameters.txt`

`bootparams(7)` man page

**Practice Exercise****Modifying the Kernel Command Line**

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

The loopback device is used to mount files as if they were devices, which is very convenient for accessing ISO images, for example. By default, the kernel supports 8 loopback devices. Modify your remote server's kernel command line so that 32 are supported instead.

Use your serverX for the following steps.

1. List the loopback devices in the /dev directory. (Loopback devices are all named *loop\**.) How many are there? \_\_\_\_\_
2. Add the parameter *max\_loop=32* to the kernel command line in **/boot/grub/grub.conf**.
3. Reboot your server.
4. Confirm the kernel booted with the modified command line.
5. List the loopback devices in the /dev directory. How many are there? \_\_\_\_\_

## Upgrading Your Kernel

### Demonstration on Upgrading a Kernel

Fill in the blanks below as your instructor discusses the following topics.

1. What (familiar) command performs a kernel update?  
\_\_\_\_\_
2. New kernels are \_\_\_\_\_, not updated.  
Because every file owned by the kernel package is versioned, or resides in a versioned directory, RPM is willing to have concurrent versions installed.
3. By default, when "updating" a kernel, **yum** will keep a total of \_\_\_\_\_ versions installed, automatically removing any older version.
4. In order to use your new kernel, you must \_\_\_\_\_ your machine.
5. While the machine will automatically reboot to your upgraded kernel, you may still choose an \_\_\_\_\_ kernel from the GRUB bootloader's menu.
6. If removing a kernel manually, you must specify not only the package name (**kernel**), but also the  
\_\_\_\_\_.



#### Warning

Do not attempt to run **yum remove kernel** without further specifying *which* kernel package to remove from the system! The command will attempt to remove *all* kernel packages installed on the system as well as all packages which depend on *kernel*. This will result in a broken and unbootable system.



## References

- Red Hat Enterprise Linux Deployment Guide
  - Chapter 23: Manually Upgrading the Kernel

**[lsmod\(8\)](#) and [modprobe\(8\)](#) man pages**

**[yum\(1\)](#), [yum.conf\(8\)](#) man pages**



## Personal Notes



## Unit Summary

### Supported Architectures and Kernel Identification

In this section you learned how to:

- Discuss supported kernel architectures and limits
- Determine which kernel is running on the system

### Managing Kernel Modules

In this section you learned how to:

- List current modules loaded in the kernel
- Load a module and its dependencies
- Remove a module
- Persistently configure a module to load with specified parameters
- Find documentation about a module

### Specifying Kernel Boot Parameters

In this section you learned how to:

- Add a kernel command line argument that persists across reboots

### Upgrading Your Kernel

In this section you learned how to:

- Update a Red Hat compiled kernel





## **UNIT EIGHTEEN**

# **SYSTEM RECOVERY TECHNIQUES**

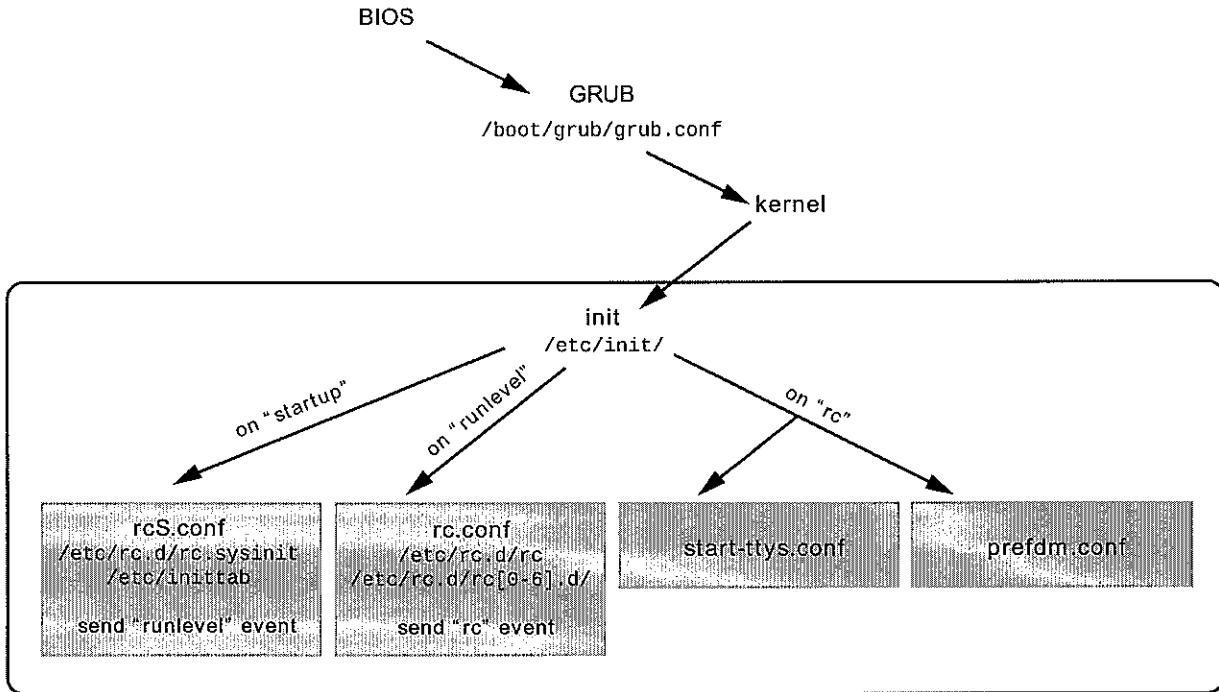
### **Introduction**

Topics covered in this unit:

- Boot Process
- Recovery Shells

# The Boot Process

Below is a simplified diagram of the Red Hat Enterprise Linux 6 boot process, from power-on to the point at which a login prompt appears on the screen.



## BIOS

The BIOS, or Basic Input/Output System, is the firmware interface built into standard x86/x86-64 hardware that puts the hardware into a known state and prepares the system to load an operating system.

### What happens?

- Detects and initializes hardware
- Determines the device to boot from

### What can go wrong?

- Incorrect or weird BIOS settings
- Bad boot device ordering

### How can it be interrupted or influenced?

- Press a vendor-specific key
- Use a vendor-specific configuration utility
- Often, <F12> can perform a one-time override of the boot ordering

## GRUB

GRUB, the GRand Unified Bootloader, is loaded by the BIOS and used to select and start the operating system, as we have already discussed.

### What happens?

- Loads initial RAM file system ("initramfs")
- Loads and executes kernel
- Provides kernel's command-line

### What can go wrong?

- Misconfiguration of the bootloader
- Bad kernel image or initramfs
- Bad kernel command line

### How can it be interrupted or influenced?

- Choose an alternate pre-configured menu item
- Use "e" or "a" to select a different kernel image or edit the kernel command-line
- Edit the kernel command-line to boot from **single** user mode
- Boot with **init=/bin/bash**

## Search & Learn: GRUB

Fill in the answers to each of the below questions, referring to available documentation, when necessary.

1. Where might we find boot oriented, and more specifically the configuration file for GRUB?
2. Where might we find documentation about what the settings in that configuration file mean?
3. What kernel will be booted by default (filename)?
4. Where is the "root" file system passed to the kernel and what is your "root" file system?
5. How does that differ from the directive in grub.conf boot entries that starts with "root"? Where is the "root" line pointing?

6. How can we restrict access to GRUB's interactive options?

## Kernel

The Linux kernel is the heart of the operating system. It is responsible for managing hardware access for userspace processes. Drivers and the KVM hypervisor are integrated parts of the kernel.

### What happens?

- Detect hardware devices
- Load device drivers (modules) for the devices



### Note

Where does the kernel get modules to load at boot time?

1. Initially, it uses the initial RAM disk configured for the kernel in **/boot/grub/grub.conf: /boot/initramfs-<VERSION>.img**
2. Once the root file system is mounted, it uses **/lib/modules/<VERSION>/**

- Mount the root file system read-only
- Start the initial process, **init**

### What can go wrong?

- Bad initial RAM file system image
- Badly identified root file system
- Corrupted root file system

### How can it be interrupted or influenced?

- Generally only through GRUB options

## init and Upstart

The first userspace process started on the machine is **/sbin/init**. The **init** process is responsible for starting all remaining userspace processes, directly or indirectly.

### What happens?

- Once the kernel is running, it starts **init**. The **init** program is responsible for completing the boot process by starting all other non-kernel system processes.

With Upstart, **init** starts "jobs" when various "events" happen, such as when the system boots, we enter a runlevel, or another **init** job starts or stops. These jobs are stored as scripts in the **/etc/init/** directory. At boot, the startup event causes **init** to run the **/etc/init/rcS.conf** job which:

- Runs **/etc/rc.d/rc.sysinit** to start LVM, mount and check file systems, set the system clock, and do other housekeeping.
- Looks in **/etc/inittab** to find the runlevel.
- Sends an event to **init** telling it to enter that runlevel.

The runlevel event causes **init** to run the **/etc/init/rc.conf** job which runs the **/etc/rc.d/rc** script with the desired runlevel as an argument:

- Example: **rc.conf** runs **rc 5**, which runs **/etc/rc.d/rc5.d/K\*** stop and **/etc/rc.d/rc5.d/S\*** start.
- The scripts are run in numeric order, first the K's and then the S's.
- The **/etc/rc.d/rc5.d/** scripts are symlinks to the scripts used by service.
- Whether the links start with a K or an S depends on whether the service has been turned **on** or **off** with **chkconfig**.

Older versions of Red Hat Enterprise Linux have a somewhat different implementation of **init** which works differently.

### What can go wrong?

- Mangled **/etc/fstab** or **/etc/crypttab**
- Network or service misconfiguration leading to hung services
- Many more...

### How can it be interrupted or influenced?

- Press "Alt-Tab" from graphical environment to view error messages
- Press "I" (capital i) during service startup to select services interactively



#### Note

Red Hat Enterprise Linux 5 and earlier used a different implementation of **init**, SysVinit, that was driven by directives in the **/etc/inittab** file. The basic boot process and scripts run by **init** were similar, however.



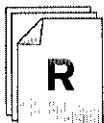
### Note

Red Hat Enterprise Linux 6 supports systems that use UEFI and the UEFI Boot Manager to load the operating system instead of a BIOS and GRUB. For more information, see the *Red Hat Enterprise Linux Installation Guide*.

## Review of Troubleshooting Symptoms

Symptoms	Possible Causes
No bootloader splash screen or prompt appears	<ul style="list-style-type: none"> <li>• GRUB is misconfigured</li> <li>• Boot sector is corrupt</li> <li>• A BIOS setting, such as disk addressing scheme, has been modified since the boot sector was written</li> </ul>
Kernel does not load at all, or loads partially before a panic occurs	<ul style="list-style-type: none"> <li>• Corrupt kernel image</li> <li>• Incorrect parameters passed to the kernel by the bootloader</li> </ul>
Kernel loads completely, but panics or fails when it tries to mount root file system and run /sbin/init	<ul style="list-style-type: none"> <li>• Bootloader is misconfigured</li> <li>• /sbin/init is corrupted or /etc/inittab is misconfigured</li> <li>• Root file system is damaged or unmountable</li> </ul>
Kernel loads completely, and /etc/rc.d/rc.sysinit is started and interrupted	<ul style="list-style-type: none"> <li>• /bin/bash is missing or corrupted</li> <li>• /etc/fstab may have an error, evident when file systems are mounted or fsck'ed</li> <li>• Corrupted non-root file systems (due to a failed fsck)</li> </ul>
Run level errors (typically services)	<ul style="list-style-type: none"> <li>• Another service required by a failing service was not configured for a given runlevel</li> <li>• Service-specific configuration errors</li> </ul>

Table18.1.Things to Check in Boot Process



## References

Red Hat Enterprise Linux Installation Guide

- Technical Appendix F: Boot Process, Init, and Shutdown

Red Hat Enterprise Linux Installation Guide

- Technical Appendix E: The GRUB Boot Loader

Red Hat Enterprise Linux Deployment Guide

- Section 23.6: Verifying the Boot Loader

Red Hat Enterprise Linux Deployment Guide

- Chapter 7: Controlling Access to Services

**init(8), event(5) and init(5) man pages**

**/etc/inittab**

**/boot/grub/grub.conf**

**info grub**



#### Practice Quiz

## Boot Process Terms

Fill in the boot process step names for each definition.

1. \_\_\_\_\_ - Firmware that runs at power on, enables features of built-in hardware and determines device to boot from.
2. \_\_\_\_\_ - Program loaded from the boot device that determines the operating system core executable to load.
3. \_\_\_\_\_ - Core operating system executable responsible for coordinating software access to hardware resources
4. \_\_\_\_\_ - First Linux process started; ultimately starts all other processes



### Practice Resequencing Exercise

## Boot Process Files

For each of the file or directory names below, write down the number of its definition from the list at the bottom.

- /boot/grub/grub.conf
- /etc/inittab
- /etc/init/
- /etc/init/rcS.conf
- /etc/rc.d/rc.sysinit
- /etc/init/rc.conf
- /etc/rc.d/rc5.d/
- /etc/init/start-ttys.conf
- /etc/init/prefdm.conf

1. Script to initialize the system (includes mounting local file systems)
2. File containing default runlevel
3. Starts 5 or 6 text based virtual consoles
4. init job responsible for starting System V based services
5. Contains bootloader configuration instructions
6. Directory containing init jobs
7. Contains links to System V services scripts to be started or stopped when entering runlevel 5
8. Starts graphical login prompt
9. init job to run system initialization script and spawn job to start System V services

## Repairing Boot Issues

At boot time, if the system has difficulty correctly mounting file systems, the boot process may be interrupted to repair the problem. In this case, the system will automatically drop to a **sulogin** shell to allow repairs as root.

The **sulogin** shell is not the same as runlevel 1 or single-user mode. At this prompt, if any file systems are mounted they are mounted read-only and the system has not fully booted to any standard runlevel.

File systems most commonly fail to mount because the file system is in an inconsistent state due to a system crash. But there are other reasons they may fail to mount. For example, the system administrator may have changed file system configuration on the system and an error or typo in **/etc/fstab** may point to a device or file system that does not exist. Another possibility is that the system administrator has attempted to mount an LVM physical volume or an encrypted device by mistake, rather than the LVM logical volume or decrypted device containing the file system.

## Repairing Damaged Filesystems

In normal operation, the kernel keeps frequently accessed file system information in memory, and only periodically commits the information to disk. If a file system becomes unexpectedly unavailable (such as due to a power outage or physical connectivity problem), the on-disk file system will contain inconsistencies. If these errors are not fixed, they are likely to lead to corrupt data.

The **fsck** command will attempt to restore a file system to a self-consistent state. The guarantee of **fsck** is not full data recovery, but consistency of the file system. On boot, the startup scripts will automatically **fsck** all file systems (if tagged in **/etc/fstab**). Minor problems will be resolved without interaction. If an automatic repair to a file system risks data loss, the boot process will drop to the **sulogin** shell to allow an administrator to run **fsck** interactively.



### Warning

Do not run **fsck** on a file system mounted read-write under any circumstances. This is likely to cause file system corruption and data loss.

1. Unmount the **/boot** file system on **/dev/vda1**.

```
[root@demo ~]# umount /dev/vda1
```

2. Check the file system on **/dev/vda1**.

```
[root@demo ~]# fsck /dev/vda1
```

3. Remount the **/boot** file system.

```
[root@demo ~]# mount /dev/vda1
```



### Important

While running, **fsck** may ask you whether or not certain changes should be made to the file system. Normally, you should answer yes to all questions, because **fsck** is asking for approval to complete a repair. You might answer no only if you are testing to see what errors **fsck** is finding or if you suspect the hardware is not responding to commands correctly, both rarer cases.



### Warning

Running **fsck** can not be undone. Running **fsck** on something that does not contain a normal file system (like an LVM physical volume) may corrupt the contents of the device, causing data loss. Eliminate other possible causes of your problem before resorting to **fsck**.

If the file system is reporting errors because of a malfunctioning storage device, **fsck** will probably not be able to solve the issue, and may introduce further errors if the device is not performing correctly. (On the other hand, in that case your only other recourse are to restore from backup to a new storage device, or to send the malfunctioning device for data recovery—in which case you may not want to run **fsck** in order to avoid further risk of data loss.)



### Workshop

## Repairing Broken fstab with slogin Shell

*Follow along with the instructor as you complete the steps below.*

You will perform these steps on your virtual servers, first breaking the **/etc/fstab** file, experiencing the resulting symptoms (TEST), then look for misconfiguration (CHECK), and finally correct the problem (FIX).

1. Execute script, **lab-setup-bootbreak-1** on serverX to break **/etc/fstab** and reboot the server.

What happens?

---

2. Enter the root password to get to slogin shell
3. View the **/etc/fstab** file (use **vim**)

What are the six fields?

Device, \_\_\_\_\_, File System Type, Mount Options, Dump Frequency, fsck Order

Hint: **fstab(5)** for details

4. Remount the root file system read-write

What command accomplishes this?

---

5. Edit **/etc/fstab** to correct the error
6. Execute **mount -a** to confirm that no errors occur
7. Reboot the system and confirm that it boots normally

The slogin shell:

- System drops into an slogin shell if it encounters problems accessing a file system in **/etc/fstab**
- At the slogin shell, the root file system is mounted read-only.
- While the slogin shell might recommend it, running **fsck** is NOT necessarily the *preferred* choice.



## References

**fsck(8), fstab(5), and sulogin(8) man pages**



Practice Quiz

## Troubleshooting the Boot Process

1. If you have file system corruption issues, the machine will boot into \_\_\_\_\_ mode.
  
2. In maintenance mode, run \_\_\_\_\_ to fix \_\_\_\_\_ corruption issues.
  
3. In maintenance mode, run \_\_\_\_\_ to mark the / partition as writable.

Test

## Criterion Test

Exercise

### Utilizing and Protecting Single User Mode

*Carefully perform the following steps. Ask your instructor if you have problems or questions.*

In this exercise, you will first recover the root password of your virtual server and then password protect GRUB to make the system more secure.

1. Run the script to scramble your root password and reboot

Run **lab-setup-bootbreak-4** from serverX

2. After verifying that you can no longer log in as root, reboot the system to reset the password back to **redhat** using single user mode.



#### Note

At the release of Red Hat Enterprise Linux 6, there was an SELinux bug which blocked the **passwd** command in single-user mode (#644820). If you have the original **selinux-policy** package installed, you must run the **setenforce 0** command in runlevel 1 before the **passwd** command for it to work. After changing the password you should run **setenforce 1** again to put SELinux back in enforcing mode.

3. Secure GRUB by adding an encrypted **password** line to the global section of **grub.conf**

What command did you use to generate the encrypted password?

---

What was the line that you added to **grub.conf**?

---

4. Reboot the system and try to get into single user mode.

Review Questions:

1. At what point were you required to enter the GRUB password?

---

2. What happens if you type in the wrong password?

---

3. Can you still boot the system without the GRUB password?
-



## Personal Notes



## Unit Summary

### The Boot Process

In this section you learned how to:

- Describe the boot process for Red Hat Enterprise Linux 6

### Repairing Boot Issues

In this section you learned how to:

- Fix problems with the boot process caused by a file system that will not mount properly
- Edit files from the read-only file system maintenance shell

---

## Appendix A. Solutions

### Automated Installation of Red Hat Enterprise Linux



#### Practice Case Study

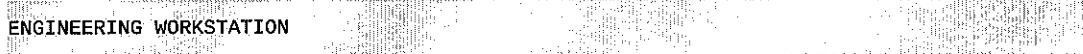
#### Creating a Kickstart File with **system-config-kickstart**

In this exercise, you will pretend that your organization is rolling out Red Hat Enterprise Linux-based workstations for its engineers, and you have been tasked with creating a Kickstart file to facilitate this.

Perform this exercise on your desktopX machine as **student**.

You will install **system-config-kickstart**. When launched, **system-config-kickstart** may ask to install a font; you may safely close this dialog. Use **system-config-kickstart** to create a Kickstart file according to the parameters specified below.

- Choose the appropriate timezone.
- The **root** password should be **redhat**
- The Kickstart should perform a fresh installation from the web server `http://instructor.example.com/pub/rhel6/dvd`
- The MBR should be cleared and the disk initialized.
- The system should have a 100 MB, **ext4** filesystem mounted at **/boot**
- The system should have a 512 MB swap partition
- All remaining disk space should be allocated to an **ext4** partition mounted at **/**
- The **eth0** device should start at boot-time and use DHCP for configuration
- Install the **Base** package set
- Have a post-installation script that adds:



to the file **/etc/issue**

- Save the Kickstart file as **/home/student/engineer.cfg**
- The resultant Kickstart file should look something like the following:



```
#platform=x86, AMD64, or Intel EM64T  
#version=DEVEL
```

```
# Firewall configuration
firewall --disabled
# Install OS instead of upgrade
install
# Use network installation
url --url="http://instructor.example.com/pub/rhel6/dvd"
# Root password
rootpw --iscrypted $1$Fjoetb6b$AmMP6QWUV/Eep5XY1nH140
# Network information
network --bootproto=dhcp --device=eth0 --onboot=on
# System authorization information
auth --useshadow --passalgo=md5
# Use graphical install
graphical
firstboot --disable
# System keyboard
keyboard us
# System language
lang en_US
# SELinux configuration
selinux --enforcing
# Installation logging level
logging --level=info
# System timezone
timezone America/New_York
# System bootloader configuration
bootloader --location=mbr
# Partition clearing information
clearpart --all --initlabel
# Disk partitioning information
part /boot --fstype="ext4" --size=100
part swap --fstype="swap" --size=1
part / --fstype="ext4" --grow --size=1

%post
echo ENGINEERING WORKSTATION >> /etc/issue
%end

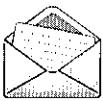
%packages
@base
%end
```

#### Practice Exercise



### Make the Kickstart File Available to Installers via HTTP

1. Login to desktopX as **student**.
2. Become **root** and deploy a web server.



#### Note

Recall that you need to Install, Start, Enable and Test

---

To verify that the Apache web server is installed, go to **System → Administration → Add/Remove Software**. Type **web server** then click **Find**. Confirm **Apache HTTP Server** is installed.

To enable and start the Apache web service, go to **System → Administration → Services**. Highlight **httpd** in the left window. Click the **Start** button, then click the **Enable** button.

To test, launch Firefox and point it to **http://desktopX** to view the test page.

3. Copy **/home/student/engineer.cfg** to **/var/www/html/**

```
[root@desktopX ~]# cp /home/student/engineer.cfg /var/www/html/
```

4. Access **http://desktopX/engineer.cfg** from a web browser to test availability.

```
[root@desktopX ~]# elinks -source http://desktopX/engineer.cfg
```

#### Practice Exercise



## Initiating a Kickstart Installation

### **Before you begin...**

Gracefully shutdown your **serverX (vserver)** virtual machine to reclaim system resources.

Use the Kickstart file created earlier (**engineer.cfg**) to deploy a new virtual machine.

1. On desktopX, create a new virtual machine using **virt-manager**. Choose **Network Boot (PXE)** as the installation method. Configure the virtual machine using the defaults if not specified below:
  - Name: engineer
  - Create a disk image on the computer's hard drive: 4 GB
2. Devise and enter an appropriate Kickstart invocation line for the **engineer.cfg** file, then start the installation.

Once the installation screen is presented from PXE, press the **Tab** key and enter the Kickstart information as below:

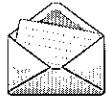
```
> vmlinuz initrd)initrd.img ks=http://desktopX/engineer.cfg
```

If the Kickstart file has errors, the installation may abort. If that is the case, edit the **/var/www/html/engineer.cfg** file on desktopX and fix the issue. Destroy the **engineer** VM and its disk and restart the installation.

#### Practice Case Study



## Modify a Kickstart File without system-config-kickstart



### Note

For time's sake you will not perform an installation using this Kickstart file.

As **root** on desktopX, create a copy of **/root/anaconda-ks.cfg** called **/home/student/projman.cfg**. Using only a text editor, modify that file so it meets the following criteria:

1. The installation must be fully automated, and exactly like the current desktopX installation (including partitioning), except...
  - The **Backup Server** package group will be installed
  - The **mtx** package, which is not installed with the **Backup Server** group by default, will be installed
  - None of the existing scripting from **%pre** and **%post** will be used
  - An **/etc/issue** file will be created in **%post**, which reads:

PROJECT MANAGEMENT

2. **ksvalidator** must be able to validate the file

When complete, copy the file to **/var/www/html/**

1. [root@desktopX]# cp /root/anaconda-ks.cfg /home/student/projman.cfg  
[root@desktopX]# chown student:student /home/student/projman.cfg  
[root@desktopX]# chmod 644 /home/student/projman.cfg
2. As student, edit the **/home/student/projman.cfg** file and include the requirements above. Make sure you uncomment the **clearpart** and other partitioning lines. It should read as follows:

```
# Kickstart file automatically generated by anaconda.

#version=RHEL6
install
url --url=ftp://instructor.example.com/pub/rhel6/dvd
lang en_US.UTF-8
keyboard us
network --device eth0 --bootproto dhcp
rootpw --iscrypted $1$YZ07ZHEP$M0u01Ut96QBgertJRauEZ/
# Reboot after installation
reboot
firewall --disabled
authconfig --useshadow --enablemd5
selinux --enforcing
timezone --utc America/New_York
```

```

bootloader --location=mbr --driveorder=sda --append="crashkernel=auto rhgb quiet"
# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
clearpart --all --drives=sda

part /boot --fstype=ext4 --size=100
part pv.LtUgsR-2QYc-f90V-xG2S-PcGm-sfYq-eHJk2i --size=28000
part swap --size=512

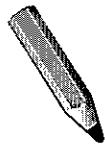
volgroup vol0 --pesize=32768 pv.LtUgsR-2QYc-f90V-xG2S-PcGm-sfYq-eHJk2i
logvol /home --fstype=ext4 --name=home --vgname=vol0 --size=500
logvol / --fstype=ext4 --name=root --vgname=vol0 --size=8192
repo --name="Red Hat Enterprise Linux" --baseurl=ftp://instructor.example.com/pub/
rhel6/dvd/ --cost=100

%packages
@Backup Server
@Base
@Console internet tools
@Core
@Desktop
@Desktop Platform
@Development Tools
@General Purpose Desktop
@Graphical Administration Tools
@Internet Browser
@Network file system client
@Printing client
@X Window System
ftp
lftp
libvirt
libvirt-client
logwatch
mtx
mutt
nss-pam-ldapd
ntp
policycoreutils-python
qemu-kvm
tigervnc
virt-manager
virt-viewer
%end

%post
echo PROJECT MANAGEMENT > /etc/issue
%end

```

3. [root@desktopX]# ksvalidator /home/student/projman.cfg
4. [root@desktopX]# cp /home/student/projman.cfg /var/www/html



Test

## Criterion Test

### Performance Checklist

#### Kickstart a Virtual Machine

##### **Before you begin...**

Shutdown the **engineer** virtual machine you created earlier and delete the virtual machine and its disk.

In the Virtual Machine Manager GUI, right-click on **engineer** and choose **Shutdown → Force Off**. Then right-click on **engineer** again and choose **Delete**. Check the **Delete associated storage files** and click **Delete**.

- Boot serverX if it is not running. Copy the **/root/anaconda-ks.cfg** file from serverX to desktopX and call it **/home/student/test.cfg**. Shutdown serverX after you have copied the file to reclaim system resources for the rest of the lab.

```
[root@serverX]# scp /root/anaconda-ks.cfg student@desktopX:test.cfg  
[root@serverX]# poweroff
```

- Modify **test.cfg** according to the following criteria:

- Add **clearpart --all** and **zerombr**, and partition storage according to the following:
  - /boot (ext4) 200 MB
  - swap 512 MB
  - / (ext4) 3 GB
- Add the **gimp** package
- Create a **/root/install-date** file with the date and time.

Add the following to the Kickstart file

```
clearpart --all  
zerombr  
part /boot --fstype=ext4 --size=200  
part swap --size=512  
part / --fstype=ext4 --size=3072  
...  
[after %packages]  
gimp  
...  
  
%post  
date > /root/install-date  
...
```

- 
- Copy **test.cfg** to **/var/www/html/** on desktopX. Make sure the file is readable by Apache.

```
[root@desktopX ~]# cp /home/student/test.cfg /var/www/html/  
[root@desktopX ~]# chmod 644 /var/www/html/test.cfg
```

- Start a virtual machine installation using your **test.cfg** Kickstart file. Name the virtual machine **test**. Use PXE as the installation method. and allocate 768 MB of RAM and 1 CPU to the virtual machine. Create a local disk of 4 GB for the virtual disk.

In Virtual Machine Manager, click on New. Enter **test** as the name, and select the Network Boot (PXE) radio button. Click Forward.

Leave the OS Type and Version as Generic and click Forward.

Change Memory (RAM) to **768** MB and click Forward.

Change the disk size to 4 GB. Click Forward.

Verify the information is correct and click Finish to begin the installation.

When the installation menu appears, press the **Tab** key and add **ks=http://desktopX/test.cfg**, then press **Enter**. The installation should proceed and reboot when complete.

- Reboot your virtual machine when it is finished installing and confirm that it installed correctly.
- After you have successfully completed the exercise and checked your work, remove this virtual machine and its associated storage to clean up.

## Accessing the Command-line



### Practice Quiz

#### Local Command-line Access

1. Type **Ctrl+Alt+F2** to switch to the second virtual terminal.
2. Type **su -** to switch to the root account.
3. Type **exit** to logout of an **su -** shell
4. Type **exit** to logout of the virtual terminal.
5. Type **Ctrl+Alt+F1** to return to the GUI.



### Practice Performance Checklist

#### Access Remote Command-line

- Login as student on your desktopX machine.
- ssh** to your serverX machine. Accept the host key if asked.

```
[student@desktopX ~]$ ssh student@serverX
The authenticity of host 'serverX (192.168.0.X+100)' can't be established. RSA key
fingerprint is 47:bf:82:cd:fa:68:06:ee:d8:83:03:1a:bb:29:14:a3.
Are you sure you want to continue connecting (yes/no)? yes
student@serverX's password: student
```

- Run the **w** command and **exit**.

```
[student@serverX ~]$ w
11:01:23 up 1 day, 19:10, 1 user, load average: 0,0,0
USER   TTY   FROM      LOGIN@ IDLE JCPU PCPU WHAT
student pts/1 desktopX 11:01  0.00s 0.12s 0.09s w
[student@serverX ~]$ exit
[student@desktopX ~]$
```

- ssh** to your serverX machine as **root**.

```
[student@desktopX ~]$ ssh root@serverX
root@serverX's password: redhat
[root@serverX ~]#
```

- Run the **w** command and exit.

```
[root@serverX ~]# w
11:01:23 up 1 day, 19:10, 1 user, load average: 0,0,0
USER   TTY   FROM      LOGIN@ IDLE JCPU PCPU
root   pts/2 desktopX 11:09  0.00s 0.13s 0.08s w
```

```
[root@serverX ~]# exit
[student@desktopX ~]$
```

- Remove the ssh known hosts file.

```
[student@desktopX ~]$ rm ~/.ssh/known_hosts
```

- ssh to serverX as root again. Accept the ssh key, login and then exit the session.

```
[student@desktopX ~]$ ssh root@serverX
The authenticity of host 'server1 (::1)' can't be established. RSA key fingerprint
is 47:bf:82:cd:fa:68:06:ee:d8:83:03:1a:bb:29:14:a3.
Are you sure you want to continue connecting (yes/no)? yes
root@serverX's password: redhat
[root@serverX ~]# exit
[student@desktopX ~]$
```

- Use ssh non-interactively to run the hostname command on serverX as root.

```
[student@desktopX ~]$ ssh root@serverX hostname
root@serverX's password: redhat
serverX.example.com
```

## Test

# Criterion Test

## Exercise

### Using the command-line

1. Switch to the second virtual terminal and login as student on your desktopX machine.

In a live-classroom environment, press **Ctrl+Alt+F2**

In a virtual-training environment, select **Ctrl+Alt+F2** from the **Send Key** menu in the upper-left of your **Classroom Systems** window.

2. Become root.

```
[student@desktopX ~]$ su -
Password: redhat
[root@desktopX ~]#
```

3. ssh to your serverX machine. Accept the host key if asked.

```
[root@desktopX ~]$ ssh serverX
The authenticity of host 'serverX (192.168.0.X+100)' can't be established.
RSA key fingerprint is 0b:3c:2c:9c:e8:03:4c:44:ec:5a:fa:b7:66:d0:c8:44.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.0.X+100' (RSA) to the list of known hosts.
```

## Appendix A. Solutions

---

```
[root@serverX ~]# password: redhat  
[root@serverX ~]#
```

- Run the **w** command and **exit**.

```
[root@serverX ~]# w  
18:36:10 up 7:15, 3 users, load average: 0.0, 0.0, 0.0  
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT  
root pts/0 192.168.0.X 17:02 00:00 0.04s 0.04s w  
[root@serverX ~]# exit  
logout  
Connection to 192.168.0.X+100 closed.  
[root@desktopX ~]#
```

- ssh to your serverX machine as **student**.

```
[root@desktopX ~]# ssh student@serverX  
student@serverX's password: student  
[student@serverX ~]$
```

- Run the **w** command and **exit**.

```
[student@serverX ~]# w  
18:36:10 up 7:15, 3 users, load average: 0.0, 0.0, 0.0  
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT  
student pts/0 192.168.0.X 17:02 0:0 0.04s 0.04s w  
[student@serverX ~]# exit  
logout  
Connection to 192.168.0.X+100 closed.  
[root@desktopX ~]#
```

- Remove the **ssh known hosts** file.

```
[root@desktopX ~]# rm -f ~/.ssh/known_hosts
```

- ssh to serverX as **student** again and note that you are prompted for the host's key again. Accept the key.

```
[root@desktopX ~]# ssh student@serverX  
The authenticity of host 'serverX (192.168.0.X+100)' can't be established.  
RSA key fingerprint is 0b:3c:2c:9c:e8:03:4c:44:ec:5a:fa:b7:66:d0:c6:44.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '192.168.0.X+100' (RSA) to the list of known hosts.  
student@serverX's password: student  
[student@serverX ~]$
```

- Once logged in, **exit** back to your shell on desktopX.

```
[student@serverX ~]# exit  
[root@desktopX ~]#
```

- Use **ssh non-interactively** to run the **ip route** command on serverX as **root**.

```
[root@desktopX ~]# ssh serverX ip route  
root@serverX's password: redhat  
.output omitted...  
[root@desktopX ~]#
```

## Intermediate Command-line Tools



### Practice Quiz

#### Comparing Hard and Symbolic Links

1. A symlink can be used across file systems.
2. A hard-link increases the link count.
3. When you create a hard link you can remove the original file without losing data.



### Practice Exercise

#### Implement Hard and Symbolic Links

Use this task to practice creating and examining hard and symbolic links.

1. Login to the desktopX machine as student.
2. Create a file named **test1** in your home directory.

```
$ echo "This file is test1" > test1
```

3. Create a symbolic link in your home directory named **symlink** pointing to the **test1** file and check that **symlink** points to the contents of **test1**.

```
$ ln -s test1 symlink  
$ cat symlink  
This file is test1
```

4. Create a hard link in your home directory named **hardlink** pointing to the **test1** file.

```
$ ln test1 hardlink
```

5. What is the link count on the **test1** file?

```
$ ls -l test1 hardlink symlink  
-rw-r--r--. 2 root root 19 Jan 26 13:35 hardlink  
lrwxrwxrwx. 1 root root 5 Jan 26 13:36 symlink -> test1  
-rw-r--r--. 2 root root 19 Jan 26 13:35 test1
```

6. Remove the **test1** file.

```
$ rm test1  
$ ls -l test1 hardlink symlink  
ls: cannot access test1: No such file or directory  
-rw-r--r--. 1 root root 19 Jan 26 13:35 hardlink  
lrwxrwxrwx. 1 root root 5 Jan 26 13:36 symlink -> test1  
$ cat symlink  
$ cat hardlink
```

```
This file is test1
```

```
$
```

7. What happened to **symlink**?

The link breaks when **test1** is removed, so **symlink** points to nothing.

8. What is the link count on **hardlink**?

The link count reduced from 2 to 1 when **test1** was removed.

9. How would you get your data back to the **test1** file?

You could either copy **hardlink** to **test1** (**cp hardlink test1**) or establish a hard link to **hardlink** (**ln hardlink test1**).

## Key tar Options

1. c = create
2. x = extract
3. t = test
4. v = verbose
5. f = file name
6. z = gzip
7. j = bzip2

### Practice Exercise

## Archive and Compress Files

Backup the **/etc** directory from serverX to a compressed tar archive file. Use the **rsync** command to copy that archive to your desktopX system. Then extract the files to the **/backups** directory on desktopX. Create the **/backups** directory if it doesn't already exist.

1. Login to serverX as **root**.

```
[root@desktopX ~]# ssh root@serverX
```

2. Create an archive of **/etc** using **gzip** compression. Save the file as **/tmp/etc.tar.gz**.

```
[root@serverX ~]# tar cvzf /tmp/etc.tar.gz /etc
```

```
[root@serverX ~]# exit
```

3. Copy the **/tmp/etc.tar.gz** file from your serverX to the **/backups** directory on your desktopX machine.

```
[root@desktopX ~]# mkdir /backups  
[root@desktopX ~]# rsync serverX:/tmp/etc.tar.gz /backups
```

4. Extract the compressed archive to **/backups** on desktopX.

```
[root@desktopX ~]# cd /backups  
[root@desktopX backups]# tar xjvf etc.tar.gz  
...
```

## Primary vim Modes

1. Command mode is used for file navigation, cut and paste, and simple commands.
2. Insert mode is used for normal text editing
3. Ex mode is used to save, quit, search and replace, and perform other complex operations.

### Practice Exercise



### Edit Files with vim

Create and edit a small text file to practice some **vim** commands. Then edit the **/etc/issues** file and see how the content of that file shows up in the login prompts of your virtual consoles.

1. Open a shell on your desktopX machine and become **root**
2. Open a practice file: **vim /tmp/testfile.txt**
3. Press **i** to go into insert mode. Then type in text so it appears as follows:

```
Change the last word in this line to goose: duck  
Quack Quack Remove the first two Quack words.  
Copy this line so it appears twice  
Make sure this is the first line in the file.
```

4. Press the **Esc** key and type **1G** to return to the first line of the file.
5. On the first line, change the word **duck** to **goose**.

Go forward a word at a time by pressing the **w** key multiple times until the cursor is on the first letter of duck. (If you go too far, press **b** to go back a word.) Then type **cw** (change word), type in **goose**, and press **Esc**.

6. On the second line, delete the first two occurrences of Quack.  
Position your cursor over the first word Quack and type **2dw** to delete both words.
7. Copy the third line, then paste it in so it appears twice.

Position the cursor over the third line (**Copy this line...**) and press **yy** to copy it. Then press **p** to paste it in after the current line. The line should now appear twice.

8. Delete the final line, then paste it so it appears as the first line in the file.

When you are done, the whole text should appear as follows:

```
Make sure this is the first line in the file.  
Change the last word in this line to goose; goose.  
Remove the first two Quack words.  
Copy this line so it appears twice  
Copy this line so it appears twice
```

Position the cursor so it is on the line that begins "Make sure this..." and press **dd** to delete the whole line. Position the cursor on the first line (type **1G**) and press **P** (that's a capital p) to paste the deleted line above the first line.

9. Save and quit the file (type :**wq**).
10. Next, use vim to edit **/etc/issue** and add the text

```
Welcome!
```

to the top of the file.

11. Test by switching to a virtual console (**Ctrl+AltF2**) and pressing **Enter** to refresh the prompt. The text **Welcome!** should appear at the top of your login prompt.

## Test

# Criterion Test

### Exercise

#### Practice using ln, tar and vim

Combine the skills you learned in this unit by creating links (hard and soft) with the **ln** command, creating a **tar** archive file, and using the **vim** text editor.

1. Login to serverX as **student**
2. Create a symlink to **/etc/passwd** in your home directory.

```
[student@serverX ~]$ ln -s /etc/passwd
```

3. Create a new file in your home directory named **newfile**.

```
[student@serverX ~]$ touch newfile
```

4. Create a hardlink to **newfile** named **newhardlink**.

```
[student@serverX ~]$ ln newfile newhardlink
```

5. Archive and compress (using bzip2 compression) your home directory to **/tmp/student-home.tar.bz2**.

```
[student@serverX ~]$ tar -jcvf /tmp/student-home.tar.bz2 ~
```

6. Create a new file in your home directory named **myvimfile.txt** using **vim**, and include the following line:

```
I wrote this using vim!
```

```
[student@serverX ~]$ vim myvimfile.txt
```

Once in **vim**, press **i** to enter Insert mode, type **I wrote this using vim!**, press **Esc** to return to Command mode, then **:wq** to enter Ex mode, save and quit.

## Regular Expressions, Pipelines and I/O Redirection

### More Regular Expression Examples

1. What does **h[aiou]t** match?  
hat, hit, hot and hut.
2. What does **b[ol][oa]t** match?  
boot, boat, blot and blat.

#### Practice Quiz



### Matching Regular Expressions

For each of the following choose all the items that match the regular expression.

1. **ba[nr]k**

*(select one or more of the following...)*

- a. bank
- b. banrk
- c. bark
- d. bakk

2. **^root**

*(select one or more of the following...)*

- a. root:x:0:0:root:/root:/bin/bash
- b. operator:x:11:0:operator:/root:/sbin/nologin
- c. root is the best!
- d. You are not root.

3. **root**

*(select one or more of the following...)*

- a. root:x:0:0:root:/root:/bin/bash
- b. operator:x:11:0:operator:/root:/sbin/nologin
- c. root is the best!
- d. You are not root.

4. **r..t**

*(select one or more of the following...)*

- a. root:x:0:0:root:/root:/bin/bash
- b. operator:x:11:0:operator:/root:/sbin/nologin
- c. root is the best!

- d. You are not root.



#### Practice Quiz

## Create a Regular Expression

For each of the following requirements, devise a regular expression that would match.

1. Line begins with "Test" or "test"

^tT]est

2. Line ends with "end."

end\.\$

3. The entire line is:

This is a test.

^This is a test\.\$

4. Any of the following names:

file5 file6 file7 file8

file[5678]

5. Any of the following names:

file2 file4 file6 file8

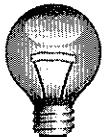
file[2468]



#### Practice Exercise

## Regular Expressions in man Pages

Open the **passwd(1)** man page and perform the following regular expression searches. Recall that you use the forward slash, /, to search man pages.



### Important

Be sure to return to the top of the man page in between each of these searches by pressing the **g** key.

1. Search for **exit**

Type: **/exit**

2. Search for a line that begins with **exit**

Type: **/^exit**

3. Search for **password**

Type: **/password**

4. Search for **password.**

Type: **/password\.**

5. Search for anything up to (and including) **pass**

Type: **/. \*pass**

#### Practice Quiz

## Create grep Commands

Devise **grep** commands to fulfill the requirements listed below based on the **/etc/passwd** file.

1. Print all usernames that begin with the letter r.

**grep '^r' /etc/passwd**

2. Print all usernames that begin with the letter g.

**grep '^g' /etc/passwd**

3. Print all accounts whose shells (last column) are **/sbin/nologin**

**grep '/sbin/nologin\$' /etc/passwd**

4. Print all accounts that have a UID or GID (third or fourth columns) of 0.

**grep ':0:' /etc/passwd**

5. Print all accounts that have a UID or GID in the range of 10-19.

**grep ':1[0123456789]:' /etc/passwd**

#### Practice Quiz

## Pipelines and Redirection

Devise commands that meet the criteria listed below

1. List all files in **/usr/share/doc** that end with the number four.

**ls /usr/share/doc | grep '4\$'**

2. Print all lines in **/etc/hosts** that have a number in them

grep '[0123456789]' /etc/hosts

3. Print the line in **/etc/hosts** that has a **127.0.0.1**

grep '127\.\.0\.\.0\.\.1' /etc/hosts

4. Run the following command as **student**, and redirect STDOUT to **/tmp/output.txt** and redirect STDERR to **/tmp/error.txt**:

find /etc -name 'host\*' > /tmp/output.txt 2> /tmp/error.txt

5. Run the following command as **student**, and redirect both STDOUT and STDERR to the **/tmp/all.txt** file.

find /etc -name 'host\*' > /tmp/all.txt 2>&1

6. Sort the **/etc/passwd** file and send it to the default printer

cat /etc/passwd | sort | lpr

7. Print out lines in **/etc/passwd** that have a three digit number between colons (:).

grep ':[0123456789][0123456789][0123456789]:' /etc/passwd

### Test

## Criterion Test

### Exercise

#### Regular expressions, pipelines and I/O redirection

1. Login to serverX as **student**
2. Print all lines in **/etc/hosts** that do not begin with a hash mark (#). [HINT: search for Invert in the grep(1) man page]. Redirect the output to **/tmp/hosts**.

[student@serverX ~]\$ grep -v '^#' /etc/hosts > /tmp/hosts

3. Append the output of **date** and **hostname** to the **/tmp/hosts** file.

[student@serverX ~]\$ date >> /tmp/hosts  
[student@serverX ~]\$ hostname >> /tmp/hosts

4. Edit **/tmp/hosts** with **gedit** or **vim** and change **127.0.0.1** to **127.0.0.2**. Comment the **date** and **hostname** output by adding a hash (#) to the beginning of the line.

The file should now look like the following:

```
192.168.0.X+100 serverX.example.com serverX # Added by NetworkManager
127.0.0.2 localhost.localdomain localhost
::1 serverX.example.com serverX localhost6.localdomain6 localhost6
#Wed Jan 26 13:46:19 EST 2011
#serverX.example.com
```

5. Using the **lpr(1)** man page, print out the lines that have the word **file** followed by zero or more of the letter **s**. Redirect that output to the **/tmp/lpr.man** file

```
[student@serverX ~]$ man lpr | grep 'files*' >/tmp/lpr.man
```

# Network Configuration and Troubleshooting



## Practice Quiz

### Viewing and Modifying Network Configuration

- Fill in the below table with the commands, utilities and filenames

<i>Setting Category</i>	<i>View Current Configuration</i>	<i>Change Configuration</i>
IP Address and Subnet Mask	<u><a href="#">ip addr</a></u>	<u>NetworkManager or /etc/sysconfig/network-scripts/ifcfg-eth*</u>
Routing/Default Gateway	<u><a href="#">ip route</a></u>	<u>NetworkManager, /etc/sysconfig/network-scripts/ifcfg-eth* or /etc/sysconfig/network</u>
System Hostname	<u><a href="#">hostname</a></u>	<u>NetworkManager, hostname or /etc/sysconfig/network</u>
Name Resolution	<u><a href="#">host</a> can be used to resolve names. /etc/hosts and /etc/resolv.conf show the current settings.</u>	<u>NetworkManager, /etc/hosts or /etc/resolv.conf</u>

Table A.1. Network Configuration from the Command-Line

2. What would a dynamic **ifcfg-eth0** contain?

**DEVICE=eth0**

**ONBOOT=yes**

**BOOTPROTO=yes**

3. What would a static **ifcfg-eth0** contain?

```
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=static
IPADDR=192.168.0.X+100
PREFIX=24
GATEWAY=192.168.0.254
DNS1=192.168.0.254
```

## Practice Exercise Document Network Settings

Before we break our network configuration, let us document the current settings on our serverX system.

1. Log in to serverX as root.
2. a. What is its current IP Address?

**192.168.0.X+100**

Found by running **ip addr**

- b. What is its current CIDR subnet mask?

**/24**

Found by running **ip addr**

- c. What is its current default gateway?

**192.168.0.254**

Found by running **ip route**

- d. What is its current hostname?

**serverX.example.com**

Found by running **hostname**

- e. What are its current DNS servers?

**192.168.0.254**

Found by running **cat /etc/resolv.conf**



Test

## Criterion Test

Exercise

### Troubleshooting Network Configuration from the Command-line

All of the following should be performed on your virtual server, serverX. You will start by running a script that will "break" your network configuration. You will have ten minutes to resolve each of the three problems. Be sure to document what you have found, as we will review at the end.

1. Run the first script to misconfigure your networking:

**lab-break-net 1**

2. Symptom: A web browser is unable to access the web page at **http://instructor.remote.test**
3. Apply the three steps: TEST, CHECK, FIX to identify and resolve the problem.
4. Document what you have found

SOLUTION: The hostname *instructor.remote.test* is not being resolved to an IP address, as */etc/resolv.conf* points to the wrong DNS server. Correcting the problem by modifying */etc/sysconfig/network-scripts/ifcfg-eth0* to have **DNS1=192.168.0.254**

5. Run the second script to misconfigure your networking:

**lab-break-net 2**

6. Symptom: A web browser is unable to access the web page at **http://instructor.remote.test**
7. Apply the three steps: TEST, CHECK, FIX to identify and resolve the problem.
8. Document what you have found

SOLUTION: The host *instructor.remote.test* is on a separate network and unreachable. **ip route** shows the default gateway pointing to the wrong router. Correcting the problem by modifying */etc/sysconfig/network-scripts/ifcfg-eth0* to have **GATEWAY=192.168.0.254**

9. Run the third script to misconfigure your networking:

**lab-break-net 3**

10. Symptom: A web browser is unable to access the web page at **http://instructor.remote.test**
11. Apply the three steps: TEST, CHECK, FIX to identify and resolve the problem.
12. Document what you have found

SOLUTION: No other host is reachable, as **ip addr** shows the IP address of the **eth0** interface is incorrect. Solve by modifying **/etc/sysconfig/network-scripts/ifcfg-eth0** to have **IPADDR=192.168.0.100+X**.

# Managing Simple Partitions and File Systems



## Practice Quiz

### Add a New File System

Fill in the blank with the command that accomplishes the task listed.

1. Identify a disk that has some free space `fdisk -cu`
2. Create a new partition on that disk `fdisk -cu /dev/device`
3. Update the kernel partition table `reboot`
4. Create a file system on the partition `mkfs -t ext4 /dev/device`
5. Determine the UUID of the file system `blkid /dev/device`
6. Create a mount point `mkdir /directory`
7. Add an entry to the file system table file Add an entry to `/etc/fstab` like the following:  
`UUID=cb79b7d0-dc14-4402-8465-6857346c9a53 /directory ext4 defaults 1 2`
8. Mount the file system `mount -a`



## Practice Resequencing Exercise

### Create Encrypted File System

For each of the file or directory names below, write down the number of its definition from the list at the bottom.

- 1 Create a new partition
- 4 Create an **ext4** file system
- 2 Format the new partition for encryption
- 6 Mount the file system on the unlocked device
- 8 Create an entry in `/etc/fstab`
- 5 Create a directory to use as a mount point
- 3 Unlock the encrypted partition
- 7 Create an entry in `/etc/crypttab`
- 9 Make LUKS aware of the password file

1. **fdisk**
2. **cryptsetup luksFormat /dev/vdaN**
3. **cryptsetup luksOpen /dev/vdaNsecret**
4. **mkfs -t ext4 /dev/mapper/secret**
5. **mkdir /secret**
6. **mount /dev/mapper/secret /secret**

7. ***secret /dev/vdaN/password/file***
8. ***/dev/mapper/secret /secret ext4 defaults 1 2***
9. ***cryptsetup luksAddKey /dev/vdaN/password/file***

 Practice Exercise

## Create and Use a New Swap Partition

Perform the following steps on serverX unless directed otherwise.

1. Start **fdisk** and create a new 256 MB partition. Change the partition type to **swap**.



### Important

To have room for creating additional partitions in the future, if needed, be sure to create an Extended partition beforehand

```
[root@serverX ~]# fdisk -cu /dev/vda
Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
p
Partition number (1-4): 3
First sector (9914368-12582911, default 9914368): Enter
Using default value 9914368
Last sector, +sectors or +size{K,M,G} (9914368-12582911, default 12582911): +256M

Command (m for help): t
Partition number (1-4): 3
Hex code (type L to list codes): 82
Changed system type of partition 3 to 82 (Linux swap / Solaris)

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
[root@serverX ~]# reboot
```

2. Prepare the new partition for use as swap

```
[root@serverX ~]# mkswap /dev/vda3
```

3. Determine the UUID

```
[root@serverX ~]# blkid /dev/vda3  
/dev/vda3: UUID="36fbc448-b969-4dcb-b940-acd084eae6bb" TYPE="swap"
```

4. Add the new partition to **/etc/fstab**

Add a line like to following to **/etc/fstab**, using the UUID value from the **blkid** command output above:

```
UUID="36fbc448-b969-4dcb-b940-acd084eae6bb" swap swap defaults 0 0
```

5. Determine current amount of swap

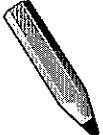
```
[root@serverX ~]# swapon -s  
Filename                                Type      Size    Used   Priority  
/dev/dm-0                                 partition 557048  0       -1
```

6. Activate the new swap

```
[root@serverX ~]# swapon -a  
swapon: /dev/mapper/vgsrv-swap: swapon failed: Device or resource busy
```

7. Verify newly activated swap

```
[root@serverX ~]# swapon -s  
Filename                                Type      Size    Used   Priority  
/dev/dm-0                                 partition 557048  0       -1  
/dev/vda3                                partition 262136  0       -2
```

Test

## Criterion Test

### Case Study

### Managing Simple Partitions and File Systems

*Before you begin...*

Run **lab-setup-storage** on desktopX to prepare serverX for the exercise.

Perform the following steps on serverX unless directed otherwise.

Your department wants to use some unallocated storage on the servers. Create additions to your system according to the following list:

- Create a new partition and **ext4** file system that is 400 MB in size. The file system should persistently mount under **/data**.
- Persistently add a swap partition that is 200 MB in size.

- Create an encrypted device with an **ext4** file system that is 256 MB in size and uses the password **testing123**. The system should prompt for the password at boot and mount the file system to **/test**.

When you are ready to check your work, run **lab-grade-storage** on serverX.



### Important

To have room for creating additional partitions in the future, if needed, be sure to create an Extended partition beforehand.

- [root@desktopX ~]# lab-setup-storage**
- Use **fdisk** to add 3 partitions, for the standard file system, the swap partition, and the encrypted file system, respectively.
  - Start an interactive session with **fdisk**, and print the existing table.

```
[root@serverX ~]# fdisk -cu /dev/vda
Command (m for help): p
Disk /dev/vda: 21.5 GB, 21474836480 bytes
...
Device Boot Start End Blocks Id System
/dev/vda1 * 2048 526335 262144 83 Linux
/dev/vda2 526336 9914367 4694016 8e Linux LVM
```

- In order to avoid later problems caused by a limited number of primary partitions, create an extended partition which spans the remainder of the disk.

```
Command (m for help): n
Command action
  e extended
  p primary partition (1-4)
e
Partition number (1-4): 3
First sector (9914368-41943039, default 9914368): Enter
Using default value 9914368
Last sector, +sectors or +size[K,M,G] (9914368-41943039, default 41943039): Enter
Using default value 41943039
Command (m for help): p
Disk /dev/vda: 21.5 GB, 21474836480 bytes
...
Device Boot Start End Blocks Id System
/dev/vda1 * 2048 526335 262144 83 Linux
/dev/vda2 526336 9914367 4694016 8e Linux LVM
/dev/vda3 9914368 41943039 16014336 5 Extended
```

- c. Add a 400 MB partition for the **ext4** file system.

```
Command (m for help): n
Command action
  l  logical (5 or over)
  p  primary partition (1-4)
l
First sector (9916416-41943039, default 9916416): Enter
Using default value 9916416
Last sector, +sectors or +size{K,M,G} (9916416-41943039, default 41943039): +400M
```

- d. Add a 200 MB partition, and adjust its label appropriately for a swap partition.

```
Command (m for help): n
Command action
  l  logical (5 or over)
  p  primary partition (1-4)
l
First sector (10737664-41943039, default 10737664): Enter
Using default value 10737664
Last sector, +sectors or +size{K,M,G} (10737664-41943039, default 41943039): +200M
Command (m for help): t
Partition number (1-6): 6
Hex code (type L to list codes): 82
Changed system type of partition 6 to 82 (Linux swap / Solaris)
```

- e. Add a 256 MB partition for the encrypted **ext4** partition.

```
Command (m for help): n
Command action
  l  logical (5 or over)
  p  primary partition (1-4)
l
First sector (11149312-41943039, default 11149312): Enter
Using default value 11149312
Last sector, +sectors or +size{K,M,G} (11149312-41943039, default 41943039): +256M
```

- f. Admire your work, and exit committing changes.

```
Command (m for help): p
Disk /dev/vda: 21.5 GB, 21474836480 bytes
...
      Device Boot   Start     End   Blocks Id System
/dev/vda1    *     2048   526335   262144  83 Linux
/dev/vda2        526336  9914367  4694016  8e Linux LVM
/dev/vda3     9914368  41943039 16014336    5 Extended
/dev/vda5     9916416  10735615  409600  83 Linux
/dev/vda6    10737664  11147263  204800  82 Linux swap / Solaris
/dev/vda7    11149312  11673599  262144  83 Linux

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
```

```
WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
```

- g. Noting the warning, confirm that the kernel is not yet aware of the newly created disk partitions.

```
[root@serverX ~]# cat /proc/partitions
major minor #blocks name
8       0   20971520 vda
8       1    262144 vda1
8       2    4694016 vda2
253     0    557056 dm-0
253     1   3473408 dm-1
253     2    262144 dm-2
```

- h. Reboot serverX.  
i. Once serverX has rebooted, confirm that the kernel is now aware of the new partitions.

```
[root@serverX ~]# cat /proc/partitions
major minor #blocks name
8       0   20971520 vda
8       1    262144 vda1
8       2    4694016 vda2
8       3      1 vda3
8       5   409600 vda5
8       6   204800 vda6
8       7    262144 vda7
253     0    557056 dm-0
253     1   3473408 dm-1
253     2    262144 dm-2
```

3. Format the **ext4** partition, decide upon a mount point (we will use **/data**), and configure the system to automatically mount the partition on bootup.

- a. Lay down the **ext4** file system on the 400MB partition, and determine the resulting UUID.

```
[root@serverX ~]# mkfs.ext4 /dev/vda5
mke2fs 1.41.12 (17-May-2010)
Filesystem label=
...
[root@serverX ~]# blkid /dev/vda5
/dev/vda5: UUID="b06c6e49-c056-4af0-a6e1-ee79602f5bf8" TYPE="ext4"
```

- b. Add an entry (line) to **/etc/fstab**, which associates the file system (identified by UUID) with the intended mount point.

```
UUID=b06c6e49-c056-4af0-a6e1-ee79602f5bf8 /data ext4 defaults 1 2
```

- c. Create the mount point (directory), and use the **mount** command to mount all defined mount points (thereby confirming the correctness of your newly created **/etc/fstab** entry).

```
[root@serverX ~]# mkdir /data
[root@serverX ~]# mount -a
```

- d. Use the **df** command to confirm the newly created file system is mounted appropriately.

```
[root@serverX ~]# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/vgsrv-root  3.3G  2.1G  1.1G  67% /
tmpfs          246M   88K  246M   1% /dev/shm
/dev/vda1       248M   30M  206M  13% /boot
/dev/mapper/vgsrv-home  248M   11M  226M   5% /home
/dev/vda5       388M   11M  358M   3% /data
```

4. Initialize the swap partition, and arrange for it to be activated on bootup.

- a. Initialize the swap partition.

```
[root@serverX ~]# mkswap /dev/vda6
Setting up swapspace version 1, size = 204796 KiB
no label, UUID=c3f56fc4-cd69-48fe-bd87-fef41a1db3ae
```

- b. Add an entry (line) to **/etc/fstab**, which identifies the swap partition by UUID.

```
UUID=c3f56fc4-cd69-48fe-bd87-fef41a1db3ae swap swap defaults 0 0
```

- c. Activate the partition, noting active swap partitions before and after.

```
[root@serverX ~]# cat /proc/swaps
Filename           Type  Size  Used  Priority
/dev/dm-0          partition    557048  0     -1
[root@serverX ~]# swapon -a
swapon: /dev/mapper/vgsrv-swap: swapon failed: Device or resource busy
[root@serverX ~]# cat /proc/swaps
Filename           Type  Size  Used  Priority
/dev/dm-0          partition    557048  0     -1
/dev/vda6          partition  204792  0     -2
```

(Note the "busy" complaint is referring to the already active swap partition.)

5. Create and configure the encrypted partition.

- a. Scrub the partition with random data. (In lab, this step can take a significant amount of time, and can be safely skipped.)

```
[root@serverX ~]# cat /dev/urandom > /dev/vda7
( ... significant delay ... )
```

```
cat: write error: No space left on device
```

- b. Initialize the LUKS encryption layer.

```
[root@serverX ~]# cryptsetup luksFormat /dev/vda7
```

```
WARNING!
```

```
=====
```

```
This will overwrite data on /dev/vda7 irreversibly.
```

```
Are you sure? (Type uppercase yes): YES
```

```
Enter LUKS passphrase: testing123
```

```
Verify passphrase: testing123
```

- c. Open the LUKS device, choosing an arbitrary name for accessing the plaintext layer (we will use `test_plaintext`)

```
[root@serverX ~]# cryptsetup luksOpen /dev/vda7 test_plaintext
```

```
Enter passphrase for /dev/vda7: testing123
```

- d. Configure the `/etc/crypttab` file to automatically open the device on bootup, prompting the user for a password.

```
[root@serverX ~]# echo "test_plaintext /dev/vda7" >> /etc/crypttab
```

- e. Create the `ext4` file system.

```
[root@serverX ~]# mkfs.ext4 /dev/mapper/test_plaintext
```

```
mke2fs 1.41.12 (17-May-2010)
```

```
.
```

```
[root@serverX ~]# blkid /dev/mapper/test_plaintext
```

```
/dev/mapper/test_plaintext: UUID="5bf2d39e-cb79-4f4a-a276-2a306ad506c1"
```

```
TYPE="ext4"
```

- f. Establish a mount point (we will use `/test`) and add a line similar to the following to `/etc/fstab`.

```
UUID=5bf2d39e-cb79-4f4a-a276-2a306ad506c1 /test ext4 defaults 1 2
```

- g. Create the mount point, mount all partitions, and confirm that the encrypted file system was mounted.

```
[root@serverX ~]# mkdir /test
```

```
[root@serverX ~]# mount -a
```

```
[root@serverX ~]# df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/mapper/test_plaintext	246M	6.1M	228M	3%	/test

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/mapper/test_plaintext	246M	6.1M	228M	3%	/test

6. [root@serverX ~]# **lab-grade-storage**

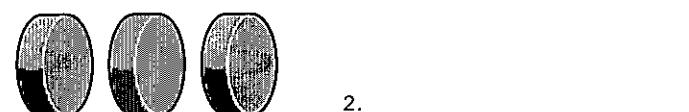
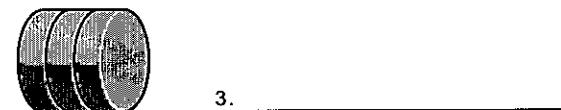
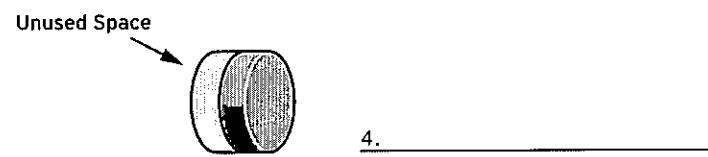
# Managing Flexible Storage with Logical Volume Manager

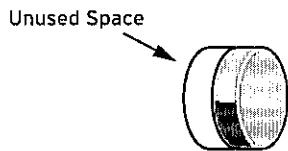


Practice Quiz

## LVM Components

1. Fill in the following graphic with the names of the components.





4. Create logical volume (LV)



3. Create volume group (VG)



2. Create physical volume (PV)



1. Partition physical storage

2. What are the smallest pieces (chunks or blocks) of the physical volume?

Physical extents

3. What is the smallest size you could make a logical volume?

The size of a single physical extent

4. What references the physical extents of a logical volume?

Logical extents

**Practice Exercise****Implement LVM and Create a Logical Volume*****Before you begin...***

Make sure to run the **lab-setup-lvm** from your desktopX system, which will prepare your serverX system for the practice exercise.

All of these steps will be performed on serverX.

1. Create a new partition of 512 MB and prepare it for use with LVM as a Physical Volume.

**Important**

To have room for creating additional partitions in the future, if needed, be sure to create an Extended partition beforehand.

```
[root@serverX ~]# fdisk -cu /dev/vda
Command (m for help): n
Command action
  e  extended
  p  primary partition (1-4)
p
Partition number (1-4): 3
First sector (9914368-12582911, default 9914368): Enter
Using default value 9914368
Last sector, +sectors or +size{K,M,G} (9914368-12582911, default 12582911): +512M

Command (m for help): t
Partition number (1-4): 3
Hex code (type L to list codes): 8e
Changed system type of partition 3 to 8e (Linux LVM)

Command (m for help): w
The partition table has been altered!
... Output Omitted ...
[root@serverX ~]# reboot
[root@serverX ~]# pvcreate /dev/vda3
  Physical volume "/dev/vda3" successfully created
```

2. Create a Volume Group named **shazam** using the Physical Volume created in the previous step.

```
[root@serverX ~]# vgcreate shazam /dev/vda3
  Volume group "shazam" successfully created
```

3. Create and format with **ext4**, a new Logical Volume of 256 MB called **/dev/shazam/storage**.

```
[root@serverX ~]# lvcreate -n storage -L 256M shazam
  Logical volume "storage" created
[root@serverX ~]# mkfs -t ext4 /dev/shazam/storage
mke2fs 1.41.12 (17-May-2010)
... Output Omitted ...
```

4. Modify your system such that **/dev/shazam/storage** is mounted at boot time as **/storage**.

```
[root@serverX ~]# mkdir /storage
```

Add the following line to the bottom of **/etc/fstab** on serverX:

```
/dev/shazam/storage  /storage  ext4  defaults 1 2
```

## Growing a Logical Volume Basic Steps

1. Verify available space in the volume group
2. Extend the logical volume
3. Extend the file system

### Practice Exercise



## Extend a Logical Volume

All of these steps will be performed on serverX.

1. Determine the amount of free space in Volume Group **shazam**.

```
[root@serverX ~]# vgdisplay shazam
  --- Volume group ---
  VG Name          shazam
  System ID        -
  Format           lvm2
  ... Output Omitted...
  VG Size          508.00 MiB
  PE Size          4.00 MiB
  Total PE         127
  Alloc PE / Size  64 / 256.00 MiB
  Free  PE / Size  63 / 252.00 MiB
  VG UUID          accvy0-3bhi-9jk8-eg7u-44AL-ARRg-r7Uo30
```

2. Extend the logical volume **/dev/shazam/storage** with *half* the available extents in the volume group using command-line tools.

```
[root@serverX ~]# lvextend -l +32 /dev/shazam/storage
  Extending logical volume storage to 384.00 MiB
  Logical volume storage successfully resized
```

3. Extend the file system mounted on **/storage** using command-line tools.

```
[root@serverX ~]# resize2fs /dev/shazam/storage
resize2fs 1.41.12 (17-May-2010)
Filesystem at /dev/shazam/storage is mounted on /storage; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 2
Performing an on-line resize of /dev/shazam/storage to 393216 (1k) blocks.
The filesystem on /dev/shazam/storage is now 393216 blocks long.
```



### Practice Exercise

## Extend a Volume Group

All of these steps will be performed on serverX.

1. Create a new 512 MB partition and prepare it for use with LVM as a Physical Volume.



### Important

To have room for creating additional partitions in the future, if needed, be sure to create an Extended partition beforehand.

```
[root@serverX ~]# fdisk -cu /dev/vda

Command (m for help): n
Command action
  e   extended
  p   primary partition (1-4)
e
Selected partition 4
First sector (10962944-12582911, default 10962944): Enter
Using default value 10962944
Last sector, +sectors or +size{K,M,G} (10962944-12582911, default 12582911): Enter
Using default value 12582911

Command (m for help): n
First sector (10964992-12582911, default 10964992): Enter
Using default value 10964992
Last sector, +sectors or +size{K,M,G} (10964992-12582911, default 12582911): +512M

Command (m for help): t
Partition number (1-5): 5
Hex code (type L to list codes): 8e
Changed system type of partition 5 to 8e (Linux LVM)

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
```

```
Syncing disks.  
[root@serverX ~]# reboot  
[root@serverX ~]# pvcreate /dev/vda5  
Physical volume "/dev/vda5" successfully created
```

2. Extend the Volume Group **shazam** by adding the Physical Volume created in the previous step.

Use **vgextend** to extend the volume group.

```
[root@serverX ~]# vgextend shazam /dev/vda5  
Volume group "shazam" successfully extended
```

## Determining Snapshot Size

1. Expected rate of change
2. Required snapshot time

### Practice Exercise

## Creating an LVM Snapshot

Compare the contents of our existing logical volume, **/dev/shazam/storage**, to a new snapshot volume, **/dev/shazam/storagesnap**, while making changes to the original volume.

All of these steps will be performed on serverX.

1. Copy the file **/usr/share/dict/linux.words** to **/storage** so you have some data to compare.

```
[root@serverX ~]# cp /usr/share/dict/linux.words /storage
```

2. Create a new 20 MB snapshot logical volume of **/dev/shazam/storage** called **storagesnap**.

```
[root@serverX ~]# lvcreate -n storagesnap -L20M -s /dev/shazam/storage  
Logical volume "storagesnap" created
```

3. Manually mount **/dev/shazam/storagesnap** read only at **/storagesnap**

```
[root@serverX ~]# mkdir /storagesnap  
[root@serverX ~]# mount -o ro /dev/shazam/storagesnap /storagesnap
```

4. List the contents of **/storagesnap** and note that they are the same as **/storage**.

```
[root@serverX ~]# ls /storagesnap /storage  
/storage:  
linux.words lost+found
```

```
/storagesnap:  
linux.words lost+found
```

5. Delete the file **/storage/linux.words** and note that it still exists in **/storagesnap**.

```
[root@serverX ~]# rm /storage/linux.words  
rm: remove regular file '/storage/linux.words'? y  
[root@serverX ~]# ls ./storagesnap/storage  
/storage:  
lost+found  
  
/storagesnap:  
linux.words lost+found
```

6. Clean up: unmount **/storagesnap**, remove the directory, and delete the **storagesnap** logical volume.

```
[root@serverX ~]# umount /storagesnap  
[root@serverX ~]# rmdir /storagesnap  
[root@serverX ~]# lvremove /dev/shazam/storagesnap  
Do you really want to remove active logical volume storagesnap? [y/n]: y  
Logical volume "storagesnap" successfully removed
```

## Test

# Criterion Test

## Case Study

### LVM Case Study

#### *Before you begin...*

Run **lab-setup-lvm** on desktopX to prepare serverX for the exercise.

Allison needs to store data for her business. Her customer database is currently 256 MB in size. The data in the database changes about 10 MB per hour on a typical day. The backup software takes 10 minutes to complete a full run.

Create a new Volume Group called **allison** with enough space for both a 512 MB volume and a snapshot of that volume for the backup software.

Once the volume group is created, create within it a 512 MB logical volume for Allison's customer database called **custdb**. Also create a snapshot volume of Allison's customer database called **custdbsnap** for her backup software.

When you are ready to check your work, run **lab-grade-lvm** on serverX.

1. 

```
[root@serverX ~]# lab-setup-lvm
```
2. Create a new 1 GB partition using **fdisk**, and reboot the machine for the changes to take effect.

## Appendix A. Solutions

---

```
[root@serverX ~]# fdisk -cu /dev/vda
Command (m for help): p
Disk /dev/vda: 21.5 GB, 21474836480 bytes
...
Device Boot Start End Blocks Id System
/dev/vda1 * 2048 526335 262144 83 Linux
/dev/vda2 526336 9914367 4694016 8e Linux LVM

Command (m for help): n
Command action
e extended
p primary partition (1-4)
p
Partition number (1-4): 3
First sector (9914368-41943039, default 9914368):
Using default value 9914368
Last sector, +sectors or +size{K,M,G} (9914368-41943039, default 41943039): +1G

Command (m for help): w
...
The kernel still uses the old table. The new table will be used at
the next reboot or after you run partprobe(8) or kpartx(8)
Syncing disks.
```

Reboot serverX.

3. Prepare the newly create partition for use with LVM.

```
[root@serverX ~]# pvcreate /dev/vda3
Physical volume "/dev/vda3" successfully created
```

4. Create a new volume group called **allison** using the new partition.

```
[root@serverX ~]# vgcreate allison /dev/vda3
Volume group "allison" successfully created
```

5. Create a 512 MB logical volume for Allison's customer database.

```
[root@serverX ~]# lvcreate -n custdb -L512M allison
Logical volume "custdb" created
```

6. Create a 10 MB snapshot volume of Allison's customer database

```
[root@serverX ~]# lvcreate -n custdbsnap -L10M -s /dev/allison/custdb
Rounding up size to full physical extent 12.00 MiB
Logical volume "custdbsnap" created
```

7. When you are ready to check your work, run **lab-grade-lvm** on serverX.

```
[root@serverX ~]#lab-grade-lvm
```

## Accessing Network File Sharing Services



### Practice Exercise

#### Accessing an NFS File Server

These steps should be performed on serverX.

After verifying the availability of an NFS share, create the necessary mountpoint and temporarily mount the share.

1. Verify that **/var/ftp/pub** is an available NFS share on the server **instructor.example.com**.

```
showmount -e instructor.example.com
```

2. Create the directory **/server** for use as a mount point.

```
mkdir /server
```

3. Devise and execute a command to mount the NFS share to the mount point.

```
mount instructor.example.com:/var/ftp/pub /server
```

4. List the contents of the mount point to verify that it is the contents of the NFS share from **instructor.example.com**

```
ls /server
```

5. Cleanup by unmounting the share

```
umount /server
```



### Practice Exercise

#### Automount NFS Share with Indirect Maps

These steps should be performed on serverX.

You will now set up the automounter to automatically mount an NFS export on a specified directory on demand.

- The NFS server is **instructor.example.com**
- The NFS export on the server is **/var/ftp/pub**
- The automatic mount point on your station should be **/server/public**
- Use an indirect map (not **/net**) to implement this
- Validate that the automount works by changing directory to the mount point and accessing the files in the share

Perform the following steps:

1. Create/modify **autofs** service configuration files.

Add the following line to **/etc/auto.master**:

```
/server /etc/auto.server
```

Create **/etc/auto.server** with the following content:

```
public -ro instructor.example.com:/var/ftp/pub
```

2. Reload the automounter.

```
service autofs reload
```

3. Access the share.

```
ls /server/public
```

Test

## Criterion Test

### Case Study

#### Automounting NFS Shares

These steps should be performed on serverX.

- Your company has taken on a new client, the Organization of Secret Hidden Undertakings (or OSHU).
- The company has a new NFS server with shares for storing files related to "special" clients: **instructor.example.com**
- The share for this client is: **/var/nfs/oshu**
- Configure your workstation such that autofs automatically mounts that share as: **/special/oshu**
- List the contents of **/special/oshu** to verify that you see the following files:

**supertopsecret.txt**

**ultrashh.txt**

1. Add the following line to **/etc/auto.master**:

```
/special /etc/auto.special
```

2. Create **/etc/auto.special** with the following contents:

## Appendix A. Solutions

---

```
 oshu -ro instructor.example.com:/var/nfs/oshu
```

3. Reload the automounter:

```
[root@serverX ~]# service autofs reload
```

# Managing User Accounts



## Practice Exercise

### Create Users Using Command-line Tools

Perform the following steps on serverX unless directed otherwise.

Create a number of users on your serverX system, setting an initial password (recording in the blanks below).

1. Log into serverX as *root*.
2. Add the user *juliet*.

```
[root@serverX ~]# useradd juliet
```

3. Confirm that *juliet* has been added using the **id** command.

```
[root@serverX ~]# id juliet  
uid=503(juliet) gid=503(juliet) groups=503(juliet)
```

4. Confirm that *juliet* has been added by examining the **/etc/passwd** file.

```
[root@serverX ~]# grep 'juliet' /etc/passwd  
juliet:x:503:503::/home/juliet:/bin/bash
```

5. Use the **passwd** command to initialize *juliet*'s password and write down the password here:

```
[root@serverX ~]# passwd juliet  
Changing password for user juliet.  
New password: juliet  
BAD PASSWORD: it is based on a dictionary word  
BAD PASSWORD: is too simple  
Retype new password: juliet  
passwd: all authentication tokens updated successfully.
```

6. Continue adding the remaining users from the list below Remember to set an initial password and write it down next to each username:

- faraday
- jack
- kate
- james
- walt
- ben
- claire

- hugo
- elvis

You can run the pair of commands, **useradd** and **passwd**, with the argument of the individual username.

Beyond the scope of this class, but you could also "script" the operation, perhaps something like this:

```
for NAME in faraday jack kate james walt ben claire hugo elvis
do
    useradd ${NAME}
    echo "password" | passwd --stdin ${NAME}
done
```

#### Practice Quiz

### Account Maintenance

1. What command would lock **elvis**'s account?

**usermod -L elvis**

2. What command would then unlock it?

**usermod -U elvis**

3. What command would cause **elvis**'s account to expire on March 15th, 2012?

**chage -E 2012-03-15 elvis**

#### Test

### Criterion Test

#### Exercise

### Managing Password Aging Policies

Your instructor will divide you into small groups. Within each group, discuss which password aging policies would be appropriate for *professors* (who will be using the machine for a long time), *graduate students* (who will be using the machine for a few years), and *summer interns* (who will only be using the machine for the summer).

- Professors: faraday, juliet
- Graduate Students: jack, kate, james
- Summer Interns: walt, ben, claire, hugo

Assign a password policy to each group of users on serverX.

1. For each group of users (professors, grads, and interns), determine a password aging policy which would be appropriate, including:
  - Account expiration dates (if appropriate).
  - Time before passwords must be changed.
  - Time before unchanged passwords force an account to go inactive.
2. Once determined, use **chage** to implement your policy for the users added in the previous section, according to their role.

**chage -E 2010-08-31 -M 30 -I 7 username**

3. Additionally, force all users to change their password on first login.

**chage -d 0 username**

## Network User Accounts with LDAP



### Practice Quiz

#### LDAP Client Configuration

1. What seven pieces of information are typically provided by *User account information* services?  
`username:password:UID:GID:GECOS:/home/dir:shell`
2. What "other" type of information can be provided by a *network directory service*?  
Authentication method
3. What are the three pieces of information a client machine needs to be configured to get user information from an LDAP directory service?  
Server's fully-qualified hostname, Base DN, and CA certificate
4. What does the command **getent passwd ldapuser1** do? Why is this useful?  
The command **getent passwd ldapuser1** will print out the account information for the `ldapuser1` user. It is useful to find information about user accounts, such as home directories, shell, names, etc.



### Practice Case Study

#### Automounting NFS Directories

These steps should be performed on serverX.

Your company is now taking on several new clients:

1. The Organization of Secret Hidden Undertakings (or OSHU)
2. Race Along the Lake Investments, Inc. (or RALII)

Your company has a new NFS server for storing files related to these "special" clients named `instructor.example.com`, with two shares: `/var/nfs/oshu` and `/var/nfs/ralii`. The expectation is that more shares will be added as new clients are signed.

The workstations need to use `autofs` to automatically mount these shares to: `/special/oshu` and `/special/ralii`, respectively with read-only permissions.

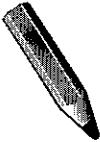
Given the expectation that additional clients will be signed shortly, implement this using `autofs` wildcards and metacharacters.

1. Modify the configuration of the automounter on serverX so that if any directory is accessed in `/special`, the automounter will mount the NFS export `instructor.example.com:/var/nfs dirname` on it if that export exists. In this example, "dirname" stands in for any possible directory name:

```
# cat /etc/auto.master  
/special /etc/auto.special
```

```
# cat /etc/auto.special  
* -ro instructor.example.com:/var/nfs/&
```

2. Reload the service with **service autofs reload**.
3. Test, by in turn accessing the contents of **/special/oshu** and **/special/ralii**.



## Test

# Criterion Test

### Case Study

#### Get Network User Information from an LDAP Directory Service

You will now configure serverX to get information about network users from an LDAP directory server available to all machines in the classroom.

Here is information that was provided to you about the LDAP server:

- Hostname: *instructor.example.com*
- Search Base DN: *dc=example,dc=com*
- CA Certificate: <http://instructor.example.com/pub/EXAMPLE-CA-CERT><sup>1</sup>

You will then configure serverX to automatically mount the home directories of your LDAP-based network users when they log in.

Here is information which was provided to you about the NFS storage server that contains the home directories:

- Hostname: *instructor.example.com*
  - Exported Directory: **/home/guests/**
1. Login to serverX as **root**. If you use **ssh**, include the **-X** option to forward graphical interfaces to your workstation.
  2. Use **system-config-authentication** to configure serverX to get network user information from the classroom LDAP server.
    1. For "User Account Database", choose **LDAP**.
      - a. For "LDAP Search Base DN:", enter *dc=example,dc=com*.
      - b. For LDAP Server:, enter *instructor.example.com*
      - c. Select the "Use TLS to encrypt connects" checkbox.
      - d. Choose "Download CA Certificate...", and enter <http://instructor.example.com/pub/example-ca.crt>.
    2. For "Authentication Method", choose **LDAP**.

3. Choose **Apply**.
3. Use the **id** command to confirm that the user **ldapuserX** (replace X with your station number) is defined on the system.

```
[root@server1 ~]# id ldapuser1  
uid=1701(ldapuser1) gid=1701(ldapuser1) groups=1701(ldapuser1)
```

4. Use the **getent** command to look up the user information for **ldapuser1**.

```
[root@server1 ~]# getent passwd ldapuser1  
ldapuser1:*:1701:1701:LDAP Test User 1:/home/guests/ldapuser1:/bin/bash
```

5. Login to serverX as **ldapuserX** or any of the other network users, using the password "password". (Do not worry if you get an error on login about a non-existent home directory, that is expected.)

```
[student@desktop1 ~]# ssh ldapuser1@server1  
ldapuser1@server1's password: password  
Could not chdir to home directory /home/guests/ldapuser1: No such file or directory  
-bash-4.1$
```

6. Modify the configuration of the automounter on serverX so that if any directory **ldapuserX** is accessed in /home/guests, the automounter will attempt to mount the equivalent NFS exported directory from **instructor.example.com:/home/guests/ldapuserX** (Hint: Use wildcard syntax.)

1. Add the following line to **/etc/auto.masters**.

```
/home/guests    /etc/auto.guests
```

2. Create the file **/etc/auto.guests**, with the following single line.

```
*      -rw,hard,intr    instructor.example.com:/home/guests/&
```

3. Restart the automounter. (Note: in early Red Hat Enterprise Linux 6 releases, the **autofs** service script has a bug, and the standard **restart** does not work.)

```
[root@server1 ~]# service autofs stop  
Stopping automount:                                [ OK ]  
[root@server1 ~]# service autofs start  
Starting automount:                                [ OK ]
```

7. Log into serverX as **IdapuserX**, where "X" is your station number, with the password "password". The user's home directory should be automatically mounted.

```
[student@desktop1 ~]$ ssh ldapuser1@server1  
The authenticity of host 'server1 (192.168.0.101)' can't be established.  
RSA key fingerprint is 33:fa:a1:3c:98:30:ff:f6:d4:99:00:4e:7f:84:3e:c3.  
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added 'server1,192.168.0.101' (RSA) to the list of known hosts.  
ldapuser1@server1's password: password  
Last login: Thu Dec 16 14:59:49 2010 from instructor.example.com  
[ldapuser1@server1 ~]$ pwd  
/home/guests/ldapuser1  
[ldapuser1@server1 ~]$ df  
Filesystem      1K-blocks    Used Available Use% Mounted on  
instructor.example.com:/home/guests/ldapuser1      1032192     36864    943104   4% /home/guests/ldapuser1
```

# Controlling Access to Files



## Practice Exercise Managing Groups

For the users that were created earlier on serverX, we will now put them into groups

<i>Group</i>	<i>groupname</i>	<i>user_list</i>
Professors	<i>profs</i>	<i>faraday, juliet, elvis</i>
Graduate Students	<i>grads</i>	<i>jack, kate, james, elvis</i>
Summer Interns	<i>interns</i>	<i>walt, ben, claire, hugo, elvis</i>

Table A.2. User and Group Assignments

Notice that there is one additional user that you will need to create named **elvis**, who should be placed in all three groups.

1. Create the groups specified in the table above, and assign the appropriate group members.

If you do not have the following user accounts, add them using **useradd username**

```
[root@serverX ~]# groupadd profs
[root@serverX ~]# usermod -aG profs faraday
[root@serverX ~]# usermod -aG profs juliet
[root@serverX ~]# usermod -aG profs elvis

[root@serverX ~]# groupadd grads
[root@serverX ~]# usermod -aG grads jack
[root@serverX ~]# usermod -aG grads kate
[root@serverX ~]# usermod -aG grads james
[root@serverX ~]# usermod -aG grads elvis

[root@serverX ~]# groupadd interns
[root@serverX ~]# usermod -aG interns walt
[root@serverX ~]# usermod -aG interns ben
[root@serverX ~]# usermod -aG interns claire
[root@serverX ~]# usermod -aG interns hugo
[root@serverX ~]# usermod -aG interns elvis
```

2. Use the **id** command to verify group memberships for all users.

```
[root@serverX ~]# id juliet
uid=503(juliet) gid=503(juliet) groups=503(juliet),513(profs)
[root@serverX ~]# id faraday
uid=504(faraday) gid=504(faraday) groups=504(faraday),513(profs)
[root@serverX ~]# id jack
uid=505(jack) gid=505(jack) groups=505(jack),514(grads)
[root@serverX ~]# id kate
uid=506(kate) gid=506(kate) groups=506(kate),514(grads)
[root@serverX ~]# id james
uid=507(james) gid=507(james) groups=507(james),514(grads)
[root@serverX ~]# id walt
uid=508(walt) gid=508(walt) groups=508(walt),515(interns)
[root@serverX ~]# id ben
uid=509(ben) gid=509(ben) groups=509(ben),515(interns)
```

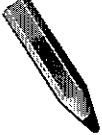
```
[root@serverX ~]# id claire
uid=510(claire) gid=510(claire) groups=510(claire),515(interns)
[root@serverX ~]# id hugo
uid=511(hugo) gid=511(hugo) groups=511(hugo),515(interns)
[root@serverX ~]# id elvis
uid=512(elvis) gid=512(elvis) groups=512(elvis),513(profs),514(grads),515(interns)
```

## Collaborative Directory Permissions

In the quiz below, use both POSIX ACLs and standard permissions, as appropriate, to solve these problems.

1. Given a normal directory, where the owning *user* has **rwx** permissions, the owning *group* has **rwx** permissions, and *other* has **---** permissions, what command would grant a second group **r-x** permissions without changing the permissions of the existing owning group or *other*?  
**setfacl -m g:group:r-x /directory**
2. What command would automatically grant that second group read-write access to any newly created regular files in that directory?  
**setfacl -m d:g:group:rw /directory**
3. *Bonus question.* What command would automatically set the owning *group* as the owning group of any newly created files in that directory?  
**chmod g+s /directory, or chmod 2770 /directory** assuming the original directory permissions as listed in the first question.

You need to ensure the directory has the set-GID permission. It turns out that there is not a way to do this with **setfacl**.



Test

## Criterion Test

### Case Study

#### Using ACLs to Grant and Limit Access

This lab uses users and groups created earlier on serverX. If you do not already have the users and groups defined, run **lab-add-users** on serverX.

Graduate students need a collaborative directory titled **/opt/research**, where they can store generated research results. Only members of the groups **profs** and **grads** should be able to create new files in the directory, and new files should have the following properties:

- The directory should be owned by user **root**.
- New files should be group owned by the group **grads**.
- Professors (members of the group **profs**) should automatically have read/write access to new files.

- Summer interns (members of the group **interns**) should automatically have read-only access to new files.
- Other users (not a member of groups **profs**, **grads**, or **interns**) should not be able to access the directory and its contents at all.

See the Solutions appendix when you are done to check your solution and to see some possible approaches.

- One possible solution is:

```
[root@serverX ~]# mkdir /opt/research
[root@serverX ~]# chgrp grads /opt/research/
[root@serverX ~]# chmod 2770 /opt/research/
[root@serverX ~]# setfacl -m g:profs:rwx /opt/research/
[root@serverX ~]# setfacl -m g:interns:rx /opt/research/
[root@serverX ~]# setfacl -m d:g:profs:rw /opt/research/
[root@serverX ~]# setfacl -m d:g:interns:r /opt/research/
```

The first three lines create **/opt/research** and make sure that it is owned by user **root**, group **grads** and that the owning user and group have and read, write, and access files, and that **grads** will own files created in the directory (set-GID is on). The next two lines grant appropriate permissions for group **profs** and group **interns** on the directory using ACLs. The next two lines grant appropriate permissions for group **profs** and group **interns** on new files created in the directory through default ACLs.



### Note

A more sophisticated and somewhat better solution is to replace the last two lines of the solution above as follows:

```
[root@serverX ~]# setfacl -m d:g:profs:rwx /opt/research/
[root@serverX ~]# setfacl -m d:g:interns:rx /opt/research/
[root@serverX ~]# setfacl -m d:o::- /opt/research/
```

By adding execute permission to the default ACLs, members of groups **profs** and **interns** will automatically be able to access newly created subdirectories of **/opt/research**. (Regular files do not get the effective execute permission automatically because the default *ACL mask* on a new regular file is **rw-**.)

The last line of the improved solution above removes all permissions from *other* for newly created files. It is not strictly necessary because the permissions on the **/opt/research** directory already blocks all access for *other* to the directory and its contents, but it is shown here as a useful example.

Use **ls** and **getfacl** to confirm your solution.

```
[root@serverX ~]# getfacl /opt/research/
getfacl: Removing leading '/' from absolute path names
# file: opt/research/
```

```
# owner: root
# group: grads
# flags: -s-
user::rwx
group::rwx
group:interns:r-x
group:prof:s:rwx
mask::rwx
other::---
default:user::rwx
default:group::rwx
default:group:interns:r--
default:group:prof:s:rwx
default:mask::rwx
[root@serverX ~]# date > /opt/research/foo
[root@serverX ~]# ls -l /opt/research/foo
-rw-rw----+ 1 root grads 29 Dec 23 12:31 /opt/research/foo
[root@serverX ~]# getfacl /opt/research/foo
getfacl: Removing leading '/' from absolute path names
# file: opt/research/foo
# owner: root
# group: grads
user::rwx
group::rwx
group:interns:r-x
group:prof:s:rwx
mask::rwx
other::---
#effective:rwx
#effective:r--
#effective:rwx
```



### Note

If you used the alternate settings above, the output of **getfacl** will show the following:

```
default:group:interns:r-x
default:group:prof:s:rwx
default:other::---
```

## Managing SELinux



### Practice Quiz

#### Basic SELinux Concepts

1. To which of the following does SELinux apply security context (check all that apply)?

*(select one or more of the following...)*

- a. Ports
- b. Processes
- c. Files
- d. Directories
- e. Remote file systems

2. SELinux can be used to:

*(select one or more of the following...)*

- a. Protect a service from running on other ports.
- b. Protect user data from applications like the web server
- c. Block remote systems from accessing local ports

- This describes a firewall.*  
d. Keep the system updated

- This describes something like Red Hat Network.*  
e. Access a web server

*This describes a web browser like Firefox.*

3. Which of the following are standard SELinux context types?

*(select one or more of the following...)*

- a. selinux\_type

- This is non-existent.*  
b. object\_r

- This is an SELinux role.*  
c. httpd\_sys\_content\_t  
d. tmp\_t  
e. user\_u

*This is an SELinux context user.*



### Practice Quiz

#### SELinux Modes

1. SELinux permissive mode allows logging, but not protection.

2. SELinux enforcing mode protects the system.
3. Which of the following are valid SELinux modes?

(select one or more of the following...)

  - a. enforcing
  - b. testing
  - c. permissive
  - d. disabled
  - e. logging

#### Practice Exercise



## Changing Enforcing and Permissive Modes

1. On serverX, change the default SELinux mode to permissive and reboot.

Modify the `/etc/sysconfig/selinux` file so the line appears as follows:

```
[root@serverX ~]# cat /etc/sysconfig/selinux
SELINUX=permissive
```

Reboot your virtual server:

```
[root@serverX ~]# init 6
```

2. After reboot, verify the system is in permissive mode.

```
[root@serverX ~]# getenforce
Permissive
```

3. Change the default SELinux mode to enforcing.

Modify the `/etc/sysconfig/selinux` file so the line appears as follows:

```
[root@serverX ~]# cat /etc/sysconfig/selinux
SELINUX=enforcing
```

4. Change the current SELinux mode to enforcing.

```
[root@serverX ~]# setenforce 1
[root@serverX ~]# getenforce
Enforcing
```



#### Practice Exercise

## Correcting SELinux File Contexts

You have been asked to adjust your remote machine's DNS configuration to exactly match the configuration from your desktop machine. You decide the easiest way is to copy the file **/etc/resolv.conf** from the local machine to the remote machine.

1. Transfer the **/etc/resolv.conf** file from your desktop machine to **root**'s home directory on serverX.

```
scp /etc/resolv.conf root@serverX:
```

2. Shell into serverX as **root**. All of the following steps should occur on your server.

3. Observe the SELinux context of the initial **/etc/resolv.conf**.

```
ls -Z /etc/resolv.conf
```

Original **/etc/resolv.conf** context: system\_u:object\_r:net\_conf\_t:s0

4. Move **resolv.conf** from **root**'s home directory to **/etc/resolv.conf**.

```
mv /root/resolv.conf /etc
```

5. Observe the SELinux context of the newly copied **/etc/resolv.conf**.

```
ls -Z /etc/resolv.conf
```

New **/etc/resolv.conf** context: unconfined\_u:object\_r:admin\_home\_t:s0

6. Restore the SELinux context of newly positioned **/etc/resolv.conf**.

```
restorecon /etc/resolv.conf
```

7. Observe the SELinux context of the restored **/etc/resolv.conf**.

```
ls -Z /etc/resolv.conf
```

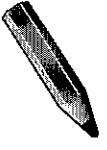
Restored **/etc/resolv.conf** context: system\_u:object\_r:net\_conf\_t:s0



#### Practice Quiz

## Monitoring SELinux Violations

1. What file contains log entries providing unique identifiers for SELinux violations? /var/log/audit/audit.log
2. Given the UUID of an SELinux violation, what command generates a text report of the problem? sealert -l UUID

Test

## Criterion Test

### Exercise

#### Managing SELinux

##### **Before you begin...**

Before you begin, run the **lab-setup-selinux** command on desktopX

1. Login to serverX as **student**. Open a terminal and switch to the **root** user.
2. Copy the *web\_content.tgz* archive from *instructor:/var/ftp/pub/materials* to */tmp*.

```
[root@serverX ~]# cp /net/instructor/var/ftp/pub/materials/web_content.tgz /tmp
```

3. Extract the archive into */tmp*.

```
[root@serverX ~]# cd /tmp  
[root@serverX tmp]# tar -xvf web_content.tgz
```

4. Move the extracted directory to */var/www/html*.

```
[root@serverX tmp]# mv web_content /var/www/html/  
[root@serverX tmp]# cd
```

5. Start the web service.

```
[root@serverX ~]# service httpd start
```

6. Try to observe the new directory with your web browser by visiting the URL *http://serverX/web\_content*.

```
[root@serverX ~]# elinks -dump http://serverX/web_content
```

7. Search your system for the UUIDs of any SELinux violations your attempt to browse the newly installed content might have generated.

```
[root@serverX ~]# cat /var/log/messages | grep 'sealert -l'
```

8. Generate text reports for the violations.

```
[root@serverX ~]# sealert -l UUID > ~/httpd_selinux.log
```

Where *UUID* is the UUID given in */var/log/messages*

9. Follow the report's advice to restore the SELinux contexts of the newly installed content.

Find the **Fix Command** section in *~/httpd\_selinux.log*

```
[root@serverX ~]# restorecon -Rv /var/www/html/web_content/
```

10. Confirm that you can view the material from your web browser by visiting the URL *http://serverX/web\_content*.

```
[root@serverX ~]# elinks -dump http://serverX/web_content
```

# Installing and Managing Software



## Practice Exercise

### Searching for and Installing Packages

Perform the following steps on serverX unless directed otherwise.

1. Attempt to run the command **gnuplot**. You should find that it is not installed.

```
[root@serverX ~]# gnuplot  
bash: gnuplot: command not found
```

2. Search for plotting packages.

```
[root@serverX ~]# yum search plot  
gnuplot.x86_64 : A program for plotting mathematical expressions and data
```

3. Find out more information about the **gnuplot** package.

```
[root@serverX ~]# yum info gnuplot  
Name        : gnuplot  
Arch       : x86_64  
...
```

4. Install the **gnuplot** package.

```
[root@serverX ~]# yum install -y gnuplot
```

5. Attempt to remove the **gnuplot** package, but say no.

```
[root@serverX ~]# yum remove gnuplot  
Removing:  
  gnuplot.x86_64          4.2.6-2.el6      @base      1.1 M  
Is this ok [y/N]: n
```

How many packages would be removed? 1

6. Attempt to remove the **gnuplot-common** package, but say no.

```
[root@serverX ~]# yum remove gnuplot-common  
Removing:  
  gnuplot-common.x86_64    4.2.6-2.el6      @base      1.3 M  
Removing for dependencies:  
  gnuplot.x86_64          4.2.6-2.el6      @base      1.1 M  
Is this ok [y/N]: n
```

How many packages would be removed? 2

## Reading Activity: YUM Component Groups

Review the **yum(1)** man page and read the sections that describe the yum commands which start with "group".

1. **yum grouplist**

Lists all available groups

2. **yum groupinfo**

Provides information about a particular group, including what packages compose the group.

3. **yum groupinstall**

Installs a group of packages.



### Note

Only package marked **mandatory** and **default** will be installed when the component group is installed.

4. **yum grouperase**

Removes a group.

5. **yum groupupdate**

Updates a group.



### Practice Exercise

## Managing YUM Component Groups

In this exercise you will gather information about the "PostgreSQL Database server" component group, and then you will install it on serverX

1. List all available component groups.

```
[root@serverX ~]# yum grouplist
```

2. Find out more information about the *PostgreSQL Database Server* component group, including a list of included packages.

```
[root@serverX ~]# yum groupinfo "PostgreSQL Database Server"
```

3. Install the *PostgreSQL Database Server* component group.

```
[root@serverX ~]# yum groupinstall "PostgreSQL Database Server"
```

#### Practice Exercise



## Handling Third-Party Software

In this exercise you will gather information about a third-party package, extract files from it, and install it as a whole on your desktopX system.

1. Download *wonderwidgets-1.0-4.x86\_64.rpm* from <http://instructor/pub/materials>.
2. What files does it contain?

```
rpm -q -p wonderwidgets-1.0-4.x86_64.rpm -l
```

3. What scripts does it contain?

```
rpm -q -p wonderwidgets-1.0-4.x86_64.rpm --scripts
```

4. How much disk space will it use when installed?

```
rpm -q -p wonderwidgets-1.0-4.x86_64.rpm -i
```

5. Use *yum localinstall* to install the package.

```
yum localinstall wonderwidgets-1.0-4.x86_64.rpm
```

#### Practice Exercise



## Using yum Repositories

You will configure your server to use a separate **yum** repository to obtain updates, and update your machine.

Perform the following steps on serverX unless directed otherwise.

1. Create the file **/etc/yum.repos.d/errata.repo**, to enable the “Updates” repository found on the instructor machine. It should access content found at the following URL: <ftp://instructor.example.com/pub/rhel6/Errata>

Create the file **/etc/yum.repos.d/errata.repo** with the following content:

```
[updates]
name=Red Hat Updates
baseurl=ftp://instructor.example.com/pub/rhel6/Errata
enabled=1
gpgcheck=1
```

2. Update all relevant software provided by the repository, using **yum update**.

```
[root@serverX ~]# yum update -y
```

Test

## Criterion Test

### Case Study

#### Update and Install Software

##### *Before you begin...*

Run **lab-setup-server** on desktopX to prepare serverX for the exercise.

You have a new server, serverX, to administrate that has very specific software requirements. It must have the latest version of the following packages installed:

**kernel** (existing package w/ an update)

**xsane-gimp** (new package)

**bzip2** (updated package)

For security reasons it should not have the **xinetd** package installed.

Do not install all updates. Only install updates for the packages listed above if they are available.

Updated packages can be found at the following URL: <ftp://instructor.example.com/pub/rhel6/Errata>

When you are ready to check your work, run **lab-grade-packages-2** on serverX.

1. [root@desktopX ~]# **lab-setup-server**
2. On serverX, create the file **/etc/yum.repos.d/updates.repo** with the following content:

```
[updates]
name=Red Hat Updates
baseurl=ftp://instructor.example.com/pub/rhel6/Errata
enabled=1
gpgcheck=1
```

3. Install the specified packages and updates, making sure to specify the individual packages, so that a full update is not performed. Note that, when packages are explicitly named on the command line, *install* and *update* are essentially synonyms.

(Do not be overly concerned if your package count is not identical to that listed below.)

```
[root@serverX ~]# yum install kernel xsane-gimp bzip2
```

```
Transaction Summary
=====
Install      15 Package(s)
Upgrade      2 Package(s)

Total download size: 39 M
Is this ok [y/N]: y

...
Importing GPG key 0xFD431D51 "Red Hat, Inc. (release key 2) <security@redhat.com>"
from /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
Is this ok [y/N]: y
Importing GPG key 0x2FA658E0 "Red Hat, Inc. (auxiliary key) <security@redhat.com>"
from /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
Is this ok [y/N]: y

...
Complete!
```

4. Make sure the **xinetd** package is removed.

```
[root@serverX ~]# yum erase xinetd
No Match for argument: xinetd
Package(s) xinetd available, but not installed.
No Packages marked for removal
```

In this case, **xinetd** was not installed.

5. [root@serverX ~]# lab-grade-packages-2

## Managing Installed Services

### Installing New Services

What command is commonly used to perform each of these steps of deploying a new service on a Red Hat Enterprise Linux system?

- **Install** the software: yum
- **Start** the service: service
- **Enable** the service at bootup: chkconfig
- **Test** the services: This will be covered on the next few pages.

## Diagnosing Network Service Problems

Perform the following steps on serverX unless directed otherwise.

```
[root@serverX ~]# service httpd restart
```

We have just restarted our **httpd** service. How can we confirm that it is working?

Try to access it using a web browser.

Assuming the restart fails, the following are checks that you can make to try to diagnose the problem. What are the commands that would show the following:

1. Is the daemon running?

If the daemon is not running, what likely next steps could you take?

- a. Examine /var/log/messages or service specific log files.
- b. Try to start the daemon "manually" from the command line, and look for any complaints.

To check that the daemon is running, run the following command:

```
[root@serverX ~]# ps aux | grep httpd
```



### Note

You may have to identify what daemons are started by the service script. For example, the **smb** service script starts **nmbd** and **smbd**.

2. Has the daemon bound to the correct ports?

If the daemon is running, but not listening to the correct port, what likely next steps could you take?

- a. Examine the service's configuration files, paying particular attention to any configuration options related to network binding.
- b. Consider your SELinux configuration. Is the daemon trying to connect to a non-standard port? Are there SELinux errors in the logs?

To check that the daemon has bound to the correct ports, run the following:

```
[root@serverX ~]# lsof -i | grep httpd
httpd    310    root    4u  IPv6  59310      0t0  TCP *:http (LISTEN)
httpd    313    apache   4u  IPv6  59310      0t0  TCP *:http (LISTEN)
```

Which ports does the service's daemons use, and are they bound to those ports? Are they bound to those ports for localhost only (127.0.0.1 or ::1) or are they bound to particular IP addresses or all IP addresses (0.0.0.0, \* or ::)?

3. Can an external client connect to the port?

If the daemon is running and bound to the correct port, but outside clients cannot connect what is the likely candidate?

- a. The problem is likely associated with kernel level firewalling.  
To verify that an external client can connect to the port, run the following on desktopX:

```
[root@serverX ~]# nc serverX 80
GET
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
```

## Test

# Criterion Test

## Exercise

### Deploying an FTP Server

Install, start, enable and test the FTP service.

Perform the following steps on serverX unless directed otherwise.

1. Install the FTP server software, which is distributed as the *vsftpd* package.

Check whether the package is already installed. If it is not, install it.

## Appendix A. Solutions

---

```
[root@serverX ~]# rpm -q vsftpd  
package vsftpd is not installed  
[root@serverX ~]# yum install -y vsftpd
```

2. Start the associated service.

Check to see whether the service is already running. If it is not, start it.

```
[root@serverX ~]# service vsftpd status  
vsftpd is stopped  
[root@serverX ~]# service vsftpd start  
Starting vsftpd for vsftpd: [ OK ]
```

3. Configure the service to start automatically at startup.

```
[root@serverX ~]# chkconfig vsftpd on
```

4. From your workstation, use the **lftp** client to connect to your server and verify that you can receive data from it.

```
[student@desktopX ~]# lftp serverX  
lftp localhost:> ls  
drwxr-xr-x 2 0 0 4096 May 26 2010 pub  
lftp server1:> quit  
[student@desktopX ~]#
```

5. Even if your service is successfully running, answer the following troubleshooting questions to confirm that the service has successfully deployed.

- Is the daemon running?

```
[root@serverX ~]# ps aux | grep vsftpd  
root 2212 0.0 0.1 51816 732 ? Ss Dec22 0:00 /usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf
```

- Has the daemon bound to the correct port (TCP port 21)?

```
[root@serverX ~]# netstat -tulpn | grep vsftpd  
tcp 0 0 0.0.0.0:21 0.0.0.0:* LISTEN 2212/vsftpd
```

- Can an external client connect to the port?

From your workstation, attempt to connect to the FTP daemon directly using the **nc** client.

```
[student@desktopX ~]# nc 192.168.0.X+100 21  
220 (vsFTPD 2.2.2)  
Ctrl+c
```

```
[student@desktopX ~]#
```

## Analyzing and Storing Logs

### Default Log Files

Fill in the name of the log file as you review the contents of `/etc/rsyslog.conf` on serverX.

1. All authentication-related messages go to /var/log/secure
2. Anything e-mail related goes to /var/log/maillog
3. Messages related to cron go to /var/log/cron
4. All other messages sent at **info** priority or higher are saved in /var/log/messages



#### Practice Quiz

### Review rsyslog

Answers the following questions:

1. Which two fields are used to match log events?  
Facility and Priority
2. What is the effect of a wildcard in the first field?  
It matches every facility
3. What is the effect of a wildcard in the second field?  
It matches every priority
4. Is it possible for the same log event to be recorded in more than one log?  
Yes, it will go to any log with matching criteria



#### Practice Performance Checklist

### Analyze a Log Summary Report

Determine the amount of free space for the root filesystem from the latest logwatch report on serverX.

- Open the email of root

```
[root@serverX ~]# mail
```

- Locate and read the most recent logwatch report. If there is no logwatch report, run the **logwatch** command to generate one manually.

```
[root@serverX ~]# mail
Heirloom Mail version 12.4 7/29/08.  Type ? for help.
"/var/spool/mail/root": 1 message 1 new
>N 1 logwatch@serverX.exa Tue Jan 1 00:25 96/2948 "Logwatch for serverX."
```

```

& Enter
(use Space to scroll to the end)

...
----- Disk Space Begin -----
Filesystem      Size   Used  Avail Use% Mounted on
/dev/mapper/vgsrv-root    3.3G  2.3G  876M  73% /
/dev/vda1        248M   30M  206M  13% /boot
/dev/mapper/vgsrv-home   248M   11M  225M   5% /home
...
----- Disk Space End -----

#####
# Logwatch End #####

```

- Record the amount of free space for the / filesystem:

In the example above, 876M.



### Practice Case Study

## Redirect Log Summary e-mails

Change the configuration of **logwatch** on serverX to send log summary reports to user **student** rather than user **root**.

*Bonus Question:* How would you send the **logwatch** reports to both **student** and **root**?

1. Modify the **/etc/logwatch/conf/logwatch.conf** file to include **MailTo=student**.
2. To send it to both **student** and **root**, change the line to:

```
MailTo = student root
```

3. Run the **logwatch** command:

```
[root@serverX]# logwatch
```

4. Check email of **student** and **root** for new reports.



### Test

## Criterion Test

### Case Study

## Log Debugging Messages

Your company wants to include a debug log on your server for analysis. Redirect all debugging level messages (and higher priority) to a file named **/var/log/debug.log**.

1. Configure **rsyslog** to create a **debug.log** in **/var/log/** directory collecting debugging messages from all services.

Add this line to **/etc/rsyslog.conf**:

```
*.debug      /var/log/debug.log
```

2. Activate the changes made to **rsyslog**.

```
[root@serverX ~]# service rsyslog restart
Shutting down system logger: [ OK ]
Starting system logger: [ OK ]
```

3. Use the **logger** command to send a message to **rsyslogd** with **debug** priority and verify the message was logged to the new log file.

```
[root@serverX ~]# logger -p debug Testing debug
[root@serverX ~]# tail /var/log/debug.log
Jan  5 11:03:04 server1 root: Testing debug
```

Note that the message will *not* show up in **/var/log/messages** if you configured and tested it correctly because **/var/log/messages** only gets messages of **info** priority and higher (not **debug**).

# Managing Processes

## Practice Exercise

### Managing Processes

**Before you begin...**

Run **lab-setup-processes** on serverX to prepare for the exercise.

```
[root@serverX ~]# lab-setup-processes
```

1. Change the priority of the process that is using the most CPU resources to 5.

```
[root@serverX ~]# top
...
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
26669 root 22 2 3828 84 12 R 78.3 0.0 0:18.41 hippo
26670 root 22 2 4008 280 156 S 21.2 0.1 0:04.94 process101
```

In **top**, press the **r** key to renice a process, then enter the PID (26669 in the example above). Press **5** to change the process to a nice value of **5**.

2. Terminate the process that is using the most memory resources.

In **top**, press the **M** key to sort by memory consumption.

```
[root@serverX ~]# top
...
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
26668 root 22 2 55032 50m 160 S 0.0 10.2 0:00.18 elephant
```

Press the **k** key to kill a process. Enter the PID (26668 in the example above). Enter the signal, or simply accept the default of **15**.

3. When you are ready to check your work, run **lab-grade-processes** on serverX.

```
[root@serverX ~]# lab-grade-processes
* Checking for terminated process... PASS
* Checking for reniced process... PASS
```

4. After you have successfully completed the exercise and checked your work, run **lab-cleanup-processes** on serverX to clean up.

```
[root@serverX ~]# lab-cleanup-processes
```

## Practice Quiz

### cronie Scheduling

1. When will the following jobs run?

## Appendix A. Solutions

---

a. 00 07 25 12 \* /usr/local/bin/open\_presents

7am Christmas Day

b. \*/5 \* \* \* \* /usr/local/bin/take\_stats

every 5 minutes

c. 07 03 \* \* \* /sbin/service xend restart

every day at 3:07 am

d. 30 16 \* \* 5 /usr/local/bin/mail\_checks

every Friday at 4:30 pm

2. Devise a cron entry which would run the script **/usr/local/bin/vacuum\_db** once a month on the first day of the month.

05 02 1 \* \* /usr/local/bin/vacuum\_db

(of course, the actual hour and minute are arbitrary.)

3. What if the machine in the previous question was down for maintenance on February first? What would be a better way to insure the database doesn't operate 2 (or more) months between vacuuming?

Write the job as a simple shell script, and drop it into **/etc/cron.monthly**:

```
#!/bin/bash  
/usr/local/bin/vacuum_db
```

## Reading Activity: The at Command

Using the **at** command, you can specify a task (referred to as a *job*) to run at a specific time in the future. The job might be a one-time backup, a check of your system, or a notification sent at a specific time. Jobs that might take a long time to complete also are good candidates for the **at** command. To do this, just use **at** to set the task to run a minute or two in the future. Then you can safely log off, since the task runs disconnected from your shell session.

The **at** command must specify when the task should run. That specification can be a specific time and/or date (**Monday, 10pm**, or **July 15**, for example). It can be relative to the current time (**now +5 min**, **now +3 days**, or **4pm +1 week**). By adding other options, you can have

mail sent when the task completes (-m) or read the task from a file (-f **file**) instead of from standard input.

After typing the **at** command line press **Enter** and proceed to type in the commands to be included in the job. The task can consist of multiple commands. When you are done typing the commands to run, press **Ctrl+d** on a line by itself to complete the task. For example:

```
[user@host ~]$ at now +2 min
at> echo "Hello from the at command" >/dev/pts/2
at> Ctrl-D <EOT>
job 1 at 2011-01-28 00:38
[user@host ~]$
```

Once an **at** job is set to run, you can list job numbers and times using the **atq** command. To see the commands included in the job, type **at -c #** (replacing # with the job number you want to view). Before a job runs, you can remove it (as long as it is your job or you are root) by typing the **atrm #** command, where # is replaced by the job number.

## Instructions

Read the **at(1)** man page focusing on the **at**, **atq**, **atrm** commands, then answer the questions below.

1. What does the **at** command do?  
The **at** command executes commands at a specified time.
2. What is the syntax of the **at** command?  
**at [OPTIONS] TIME**
3. How would you express the following time specifications to the **at** command?
  - To run a job at 4 pm three days from now?
  - To run a job at 10:00 am next July 31st?
  - To run a job at 1 am, tomorrow?  
To run a job at 4 pm three days from now, you would do **at 4pm + 3 days**  
To run a job at 10:00 am on July 31, you would do **at 10am Jul 31**  
To run a job at 1 am tomorrow, you would do **at 1am tomorrow**  
Other variations may work, if you have questions talk to your instructor.
4. What does the **atq** command do?  
The **atq** command lists the user's pending jobs, unless the user is the superuser; in that case, everybody's jobs are listed.
5. What is the format of the output of the **atq** command?  
Job number, date, hour, queue, and username.
6. What does the **atrm** command do?  
The **atrm** command deletes jobs, identified by their job number.
7. What is the syntax of the **atrm** command?

**atrm [-V] job [job...]**



## Practice Performance Checklist Scheduling Deferred Tasks

Perform the following steps on serverX unless directed otherwise.

- Determine the current system time.

```
[root@serverX ~]# date
Tue Jan 31 11:33:22 EST 2012
```

- Interactively schedule the **who** command to run 1 minute into the future.

Given the example time above, we could schedule a command like the following:

```
[root@serverX ~]# at 11:35am
at> who
at> Ctrl+d
job 1 at 2012-01-31 11:35
```

Alternately, you could use **at now +1min** to create a command that runs one minute in the future.

- While waiting for the **at** command to complete, create a simple text file containing the command **cal**. Make it executable.

```
[root@serverX ~]# echo cal > /tmp/cal-command
[root@serverX ~]# chmod 755 /tmp/cal-command
```

- Schedule the command created above to execute at teatime, both today and tomorrow.

```
[root@serverX ~]# at teatime today
at> /tmp/cal-command
at> Ctrl+d
job 2 at 2012-01-31 16:00
[root@serverX ~]# at teatime tomorrow
at> /tmp/cal-command
at> Ctrl+d
job 3 at 2012-02-01 16:00
```

- Switch places with a neighbor to perform the following two steps.

- Use **atq** to confirm the pending at jobs.

```
[root@serverX ~]# atq
2 at 2012-01-31 16:00
3 at 2012-02-01 16:00
```

- Use **atrm** to remove the job for teatime tomorrow.

Based on the output of **atq** above, job 3 is the job that should be removed.

```
[root@serverX ~]# atrm 3  
[root@serverX ~]# atq  
2 at 2012-01-31 16:00
```

- Switch back, and check your mail to confirm that the original **who** command executed.

```
[root@serverX ~]# mail  
>N 1 root Tue Jan 31 11:35 14/541 "Output from your job."  
& Enter  
From root@serverX.example.com Tue Jan 31 11:35:00 2012  
root pts/0 2012-01-31 10:40 (desktopX.example.com)  
& q
```

## Test

# Criterion Test

## Exercise

### Managing Processes

*Before you begin...*

Run **manage\_processes\_start** on serverX to prepare for the exercise.

```
[root@serverX ~]# manage_processes_start
```

Perform the following steps on serverX unless directed otherwise.

1. After running the **manage\_processes\_start** command, serverX should now be very sluggish.
2. Determine the process which is using excessive amounts of memory, and terminate the process.
  1. In a terminal, start the program **top**.
  2. Within **top**, enter "O" to bring up the sort menu.
  3. Enter **n** to sort by memory use.
  4. Note the PID of the top memory consumer.
  5. Enter **k** to kill a process.
  6. Enter the PID of the process to terminate.
  7. Hit **Enter** to choose the default signal number 15.

## Appendix A. Solutions

---

3. Determine the process which is using excessive amounts of CPU, and renice it to a niceness of 15.
  1. In a terminal, start the program **top**.
  2. Note the PIDs of the top CPU consumer.
  3. Enter **r** to renice a process.
  4. Enter the PID of the process to renice.
  5. Enter the new niceness value of 15.
4. Create a system cron job which once every half hour readjusts all processes owned by the user **elvis** to a niceness of 10.

Hint: use the **renice(1)** man page to determine the option to adjust all user processes.

Using a text editor, create the file **/etc/cron.d/nice\_elvis** with the following content:

```
# Adjust the priority of processes owned by user elvis every half hour
# to a nice value of 15.
*/30 * * * * /usr/bin/renice -n 15 -u elvis
```

5. Register a job to execute once at 3 am which runs the command **ps aux**.

```
[root@serverX ~]# at 3am
at> ps aux
at> Ctrl+d
```

# Tuning and Maintaining the Kernel



## Practice Exercise

### Loading Modules and Setting Default Parameters

Some features of the kernel-based firewall are implemented as optional kernel modules, like performing connection tracking on an ftp server.

You have been asked to load the *nf\_conntrack\_ftp* kernel module, and configure it appropriately for an FTP server listening on TCP port 21 and on 8021.

The following commands are to be run on your serverX.

1. Use the locate command to convince yourself that the *nf\_conntrack\_ftp* module is supported by your kernel.

**locate track\_ftp**

2. Load the FTP connection track module.

**modprobe nf\_conntrack\_ftp**

3. Convince yourself it's loaded.

**lsmod | grep track\_ftp**

4. Unload the FTP connection track module.

**modprobe -r nf\_conntrack\_ftp**

5. Convince yourself it's unloaded.

**lsmod | grep track\_ftp**

6. In addition to the standard port 21, you are planning to run an FTP server on the non-standard port 8021. Examine options which might let you specify non-standard ports.

**modinfo nf\_conntrack\_ftp**

7. Configure a file **/etc/modprobe.d/local.conf** which implements this option upon loading

**options nf\_conntrack\_ftp ports=21,8021**



## Practice Exercise

### Modifying the Kernel Command Line

The loopback device is used to mount files as if they were devices, which is very convenient for accessing ISO images, for example. By default, the kernel supports 8 loopback devices. Modify your remote server's kernel command line so that 32 are supported instead.

Use your serverX for the following steps.

1. List the loopback devices in the /dev directory. (Loopback devices are all named *loop\**.) How many are there? \_\_\_\_\_

**ls /dev/loop\* (8)**

2. Add the parameter *max\_loop=32* to the kernel command line in **/boot/grub/grub.conf**.
3. Reboot your server.
4. Confirm the kernel booted with the modified command line.

**cat /proc/cmdline**

5. List the loopback devices in the /dev directory. How many are there? \_\_\_\_\_

**ls /dev/loop\* (32)**

## Demonstration on Upgrading a Kernel

Fill in the blanks below as your instructor discusses the following topics.

1. What (familiar) command performs a kernel update? yum update
2. New kernels are installed, not updated. Because every file owned by the kernel package is versioned, or resides in a versioned directory, RPM is willing to have concurrent versions installed.
3. By default, when "updating" a kernel, **yum** will keep a total of 3 versions installed, automatically removing any older version.
4. In order to use your new kernel, you must reboot your machine.
5. While the machine will automatically reboot to your upgraded kernel, you may still choose an older kernel from the GRUB bootloader's menu.
6. If removing a kernel manually, you must specify not only the package name (kernel), but also the version number.

# System Recovery Techniques

## Search & Learn: GRUB

Fill in the answers to each of the below questions, referring to available documentation, when necessary.

1. Where might we find boot oriented, and more specifically the configuration file for GRUB?  
From Unit 17, the kernel is kept in /boot; grub(8) man page, /boot/grub/grub.conf
2. Where might we find documentation about what the settings in that configuration file mean?  
grub(8) does not have much detail, but points to info grub; Red Hat Deployment Guide has a section on "Configuring the GRUB boot loader"
3. What kernel will be booted by default (filename)?  
In /boot/grub/grub.conf, the default directive points to the title of a boot entry; the kernel line of the boot entry points to the kernel executable.
4. Where is the "root" file system passed to the kernel and what is your "root" file system?  
On the kernel line, the root= command-line argument identifies the device containing the root file system.
5. How does that differ from the directive in grub.conf boot entries that starts with "root"? Where is the "root" line pointing?  
The root line points to the partition that contains the files references by the kernel and initrd directives. It uses an unusual syntax where (hd0,1) is the second partition of the first hard drive detected by the BIOS.
6. How can we restrict access to GRUB's interactive options?  
We can add a password directive to the general settings at the top of the grub.conf file. It takes a cleartext password, or the hash of the password generated with grub-md5-crypt if the --md5 option is included with the password directive.

### Practice Quiz



## Boot Process Terms

Fill in the boot process step names for each definition.

1. BIOS - Firmware that runs at power on, enables features of built-in hardware and determines device to boot from.
2. GRUB - Program loaded from the boot device that determines the operating system core executable to load.
3. Kernel - Core operating system executable responsible for coordinating software access to hardware resources

4. init - First Linux process started; ultimately starts all other processes



### Practice Resequencing Exercise **Boot Process Files**

For each of the file or directory names below, write down the number of its definition from the list at the bottom.

- 5 /boot/grub/grub.conf
- 2 /etc/inittab
- 6 /etc/init/
- 9 /etc/init/rcS.conf
- 1 /etc/rc.d/rc.sysinit
- 4 /etc/init/rc.conf
- 7 /etc/rc.d/rc5.d/
- 3 /etc/init/start-ttys.conf
- 8 /etc/init/prefdm.conf

1. Script to initialize the system (includes mounting local file systems)
2. File containing default runlevel
3. Starts 5 or 6 text based virtual consoles
4. init job responsible for starting System V based services
5. Contains bootloader configuration instructions
6. Directory containing init jobs
7. Contains links to System V services scripts to be started or stopped when entering runlevel 5
8. Starts graphical login prompt
9. init job to run system initialization script and spawn job to start System V services



### Practice Quiz **Troubleshooting the Boot Process**

1. If you have file system corruption issues, the machine will boot into maintenance mode.
2. In maintenance mode, run fsck to fix file system corruption issues.
3. In maintenance mode, run mount -o remount,rw / to mark the / partition as writable.



Test

## Criterion Test

Exercise

### Utilizing and Protecting Single User Mode

In this exercise, you will first recover the root password of your virtual server and then password protect GRUB to make the system more secure.

1. Run the script to scramble your root password and reboot

Run **lab-setup-bootbreak-4** from serverX

```
[root@serverX ~]# lab-setup-bootbreak-4
```

2. After verifying that you can no longer log in as root, reboot the system to reset the password back to **redhat** using single user mode.



#### Note

At the release of Red Hat Enterprise Linux 6, there was an SELinux bug which blocked the **passwd** command in single-user mode (#644820). If you have the original **selinux-policy** package installed, you must run the **setenforce 0** command in runlevel 1 before the **passwd** command for it to work. After changing the password you should run **setenforce 1** again to put SELinux back in enforcing mode.

1. Reboot the system
2. At the "Booting Red Hat Enterprise Linux..." screen, press any key to display the GRUB menu.
3. Press **a** to append to the list of kernel arguments
4. Add **s** to the list and press **Enter** to boot into single-user mode
5. At the shell prompt, disable SELinux, reset the root password, and continue the boot process into runlevel 5.

```
[root@localhost ~]# setenforce 0
[root@localhost ~]# passwd
New Password: redhat
BAD PASSWORD: it is based on a dictionary word
BAD PASSWORD: it is too simple
Retype new password: redhat
passwd: all authentication tokens updated successfully
[root@localhost ~]# setenforce 1
[root@localhost ~]# init 5
```

3. Secure GRUB by adding an encrypted **password** line to the global section of **grub.conf**

What command did you use to generate the encrypted password?

**grub-md5-crypt**

What was the line that you added to **grub.conf**?

**password --md5 <encrypted string>**

Take care to put the **password** line *above* the first **title** line so that it will apply *globally*.

4. Reboot the system and try to get into single user mode.

Review Questions:

1. At what point were you required to enter the GRUB password?

Before we could edit/append any kernel arguments from the GRUB menu.

2. What happens if you type in the wrong password?

It continues to prompt for the password before allowing edit/append of changes.

3. Can you still boot the system without the GRUB password?

Yes, the password is only used to prevent changes to a stanza, as long as we put the password in the upper option area of **grub.conf**